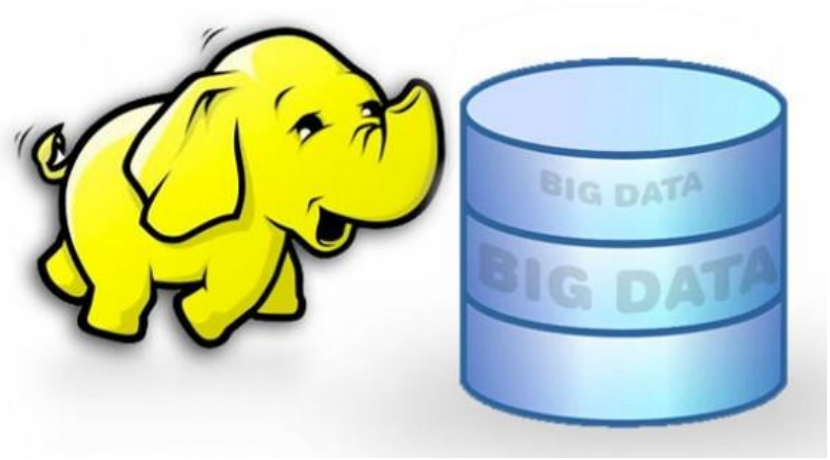
빅데이터 HDFS 개발
(HADOOP DISTRIBUTED FILE SYSTEM)



CONTENTS

- Ch01 빅데이터 개요
- Ch02 Hadoop 2.0 소개
- Ch03 Hadoop 클러스터 환경 구축
- Ch04 Hadoop 설치 및 배포
- Ch05 Hadoop 분산 클러스터 구축
- Ch06 R 프로그래밍
- Ch07 NoSQL

Chapter 01 빅데이터 개요

- 빅데이터 개요
- 빅데이터 활용
- 데이터 처리의 변화
- 구글의 빅데이터 처리 기술
- 오픈소스 하둡 프로젝트

빅데이터란?

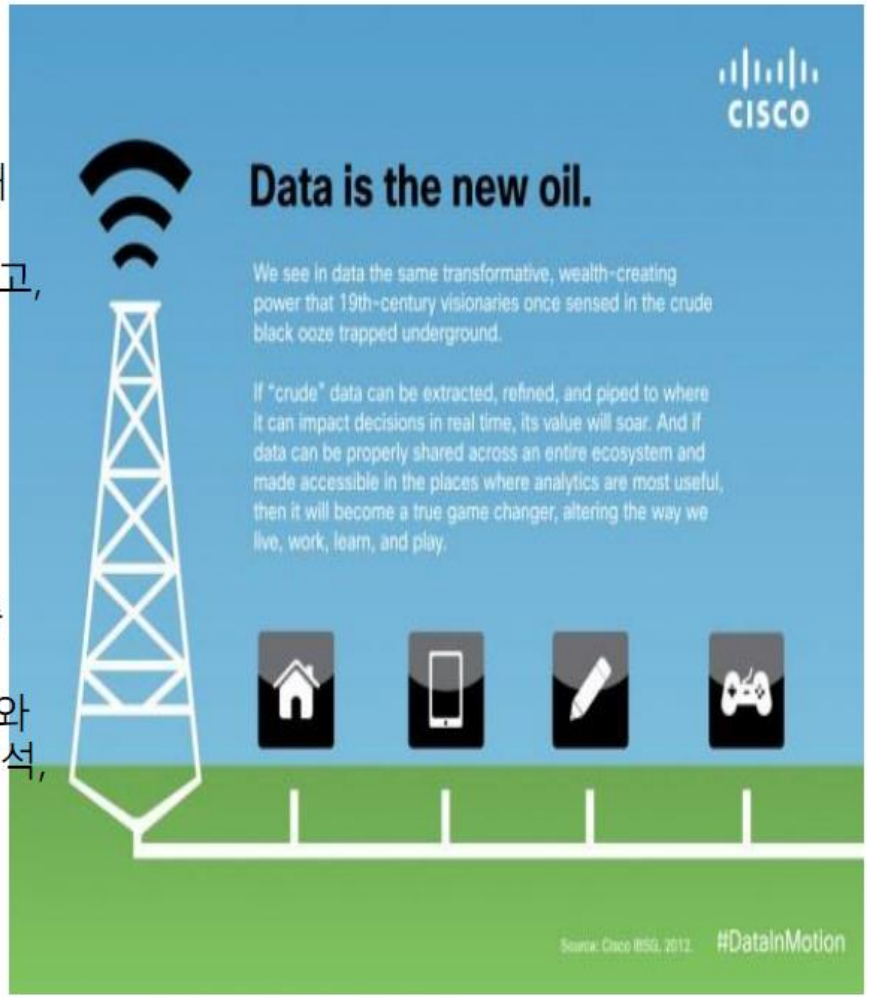
1. 디지털 환경에서 생성되는 데이터

2. 규모가 방대하고, 생성 주기도 짧고, 형태도 수치 데이터뿐 아니라 문자와 영상 데이터를 포함하는 대규모 데이터

3. 빅데이터 환경은

1) 과거에 비해 데이터의 양이 폭증
2) 데이터의 종류도 다양
3) 사람들의 행동은 물론 위치정보와 SNS를 통해 생각과 의견까지 분석, 예측가능

테라바이트(Terabate, TB)급 이상의 데이터군을 빅데이터로 통칭



빅데이터란?

형식이 다양하고 매우 빨라서 기존 방식으로는 관리, 분석이 어려운 데이터

함유근, 채승병, "빅데이타 경영을 바꾸다", 삼성경제연구소

관련 인력, 기술 등까지 포괄하는 넓은 의미로도 통용

"빅데이타란 기존의 방식으로는 관리와 분석이 매우 어려운 데이터 집합, 그리고 이를 관리•분석하기 위해 필요한 인력과 조직, 관련 기술까지 포괄하는 용어"

빅데이터 등장 배경(데이터 규모 증가)

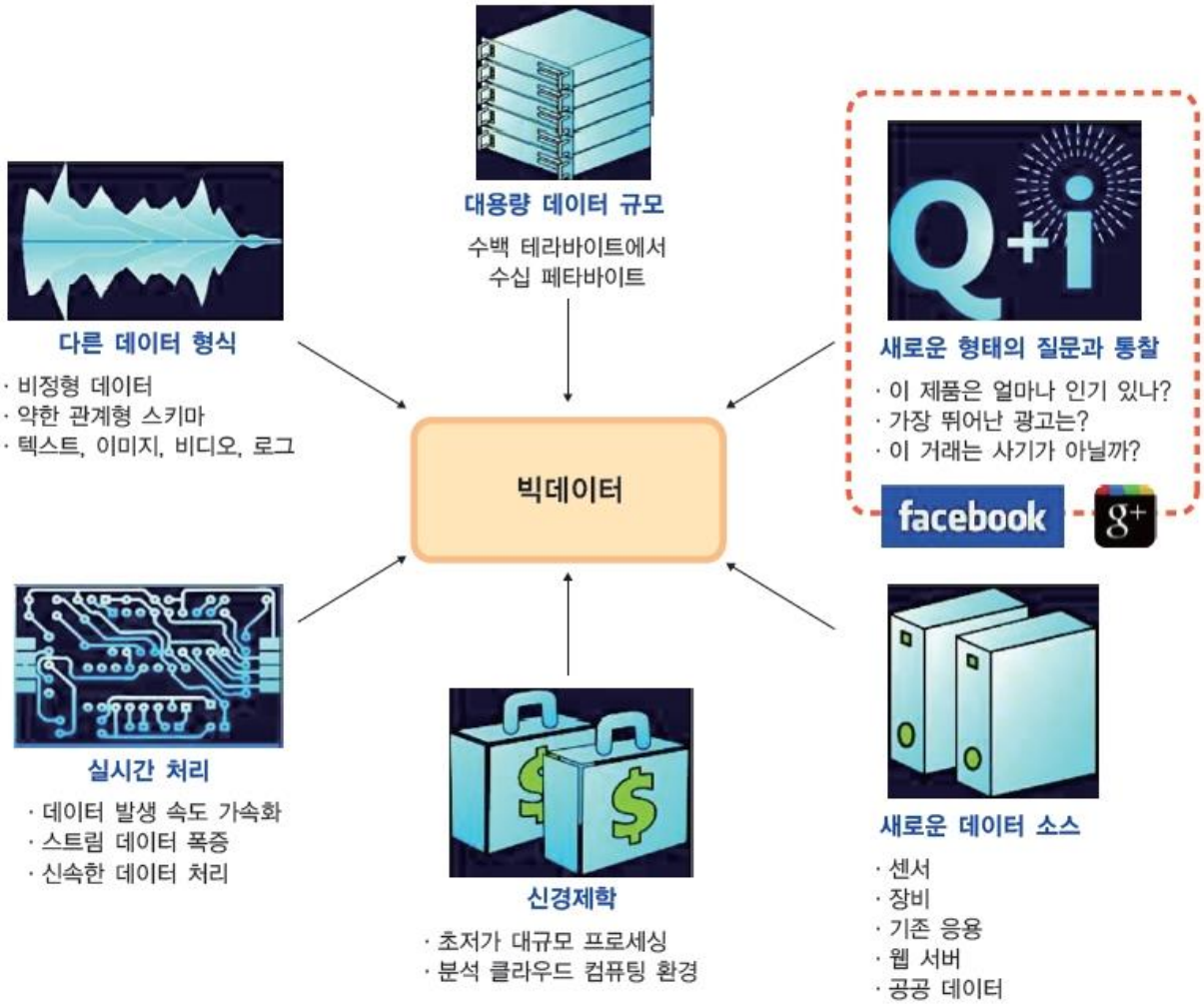
- **과거**
  상점에서 물건을 살 때만 데이터가 기록

- **현재**
  인터넷쇼핑몰에서 실제 주문은 물론 사용자의 방문기록, 관심 상품 기록, 쇼핑몰 체류시간 등이 기록되고 있다. 쇼핑몰 뿐 만이 아니라 은행, 증권과 같은 금융거래, 교육과 학습, 여가활동, 자료 검색과 이메일 등 우리의 일상생활이 데이터로 기록되고 있다. 또한 PC, 인터넷, 모바일 장치와 사물인터넷(M2M, Machine to Machine)의 확산으로 디지털 정보가 폭발적으로 증가하고 있기 때문

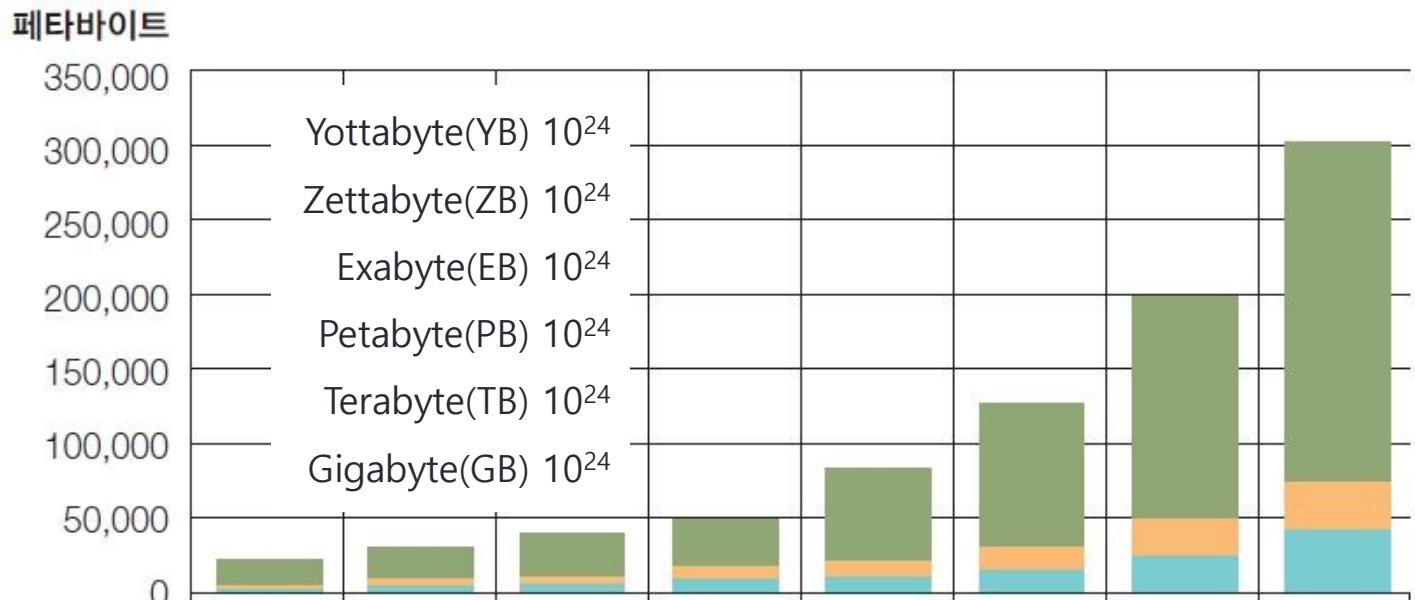
빅데이터 등장 배경(데이터 다양성 증가)

- 사용자가 직접 제작하는 UCC를 비롯한 동영상 콘텐츠
- 휴대전화와 SNS에서 생성되는 문자
- 트위터(tweeter)에서만 하루 평균 1억 5500만 건
- 유튜브(YouTube) 하루 평균 동영상 재생건수 40억회
- 주요 도로와 공공건물은 물론 심지어 아파트 엘리베이터 안에까지 설치된 CCTV가 촬영하고 있는 영상 정보

빅데이터 등장 배경(데이터 다양성 증가)



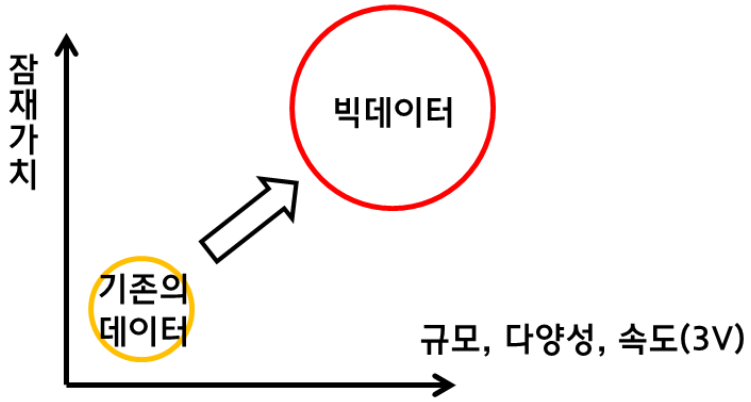
빅데이터 등장 배경(데이터 발생 속도 증가)



	2008	2009	2010	2011	2012	2013	2014	2015
비정형	11,430	16,737	25,127	39,237	59,600	92,536	147,885	226,716
정형(데이터베이스)	1,952	2,782	4,065	6,179	9,140	13,824	21,532	32,188
정형(이메일)	1,652	2,552	4,025	6,575	10,411	16,796	27,817	44,091

정형과 비정형 데이터 유형의 변화

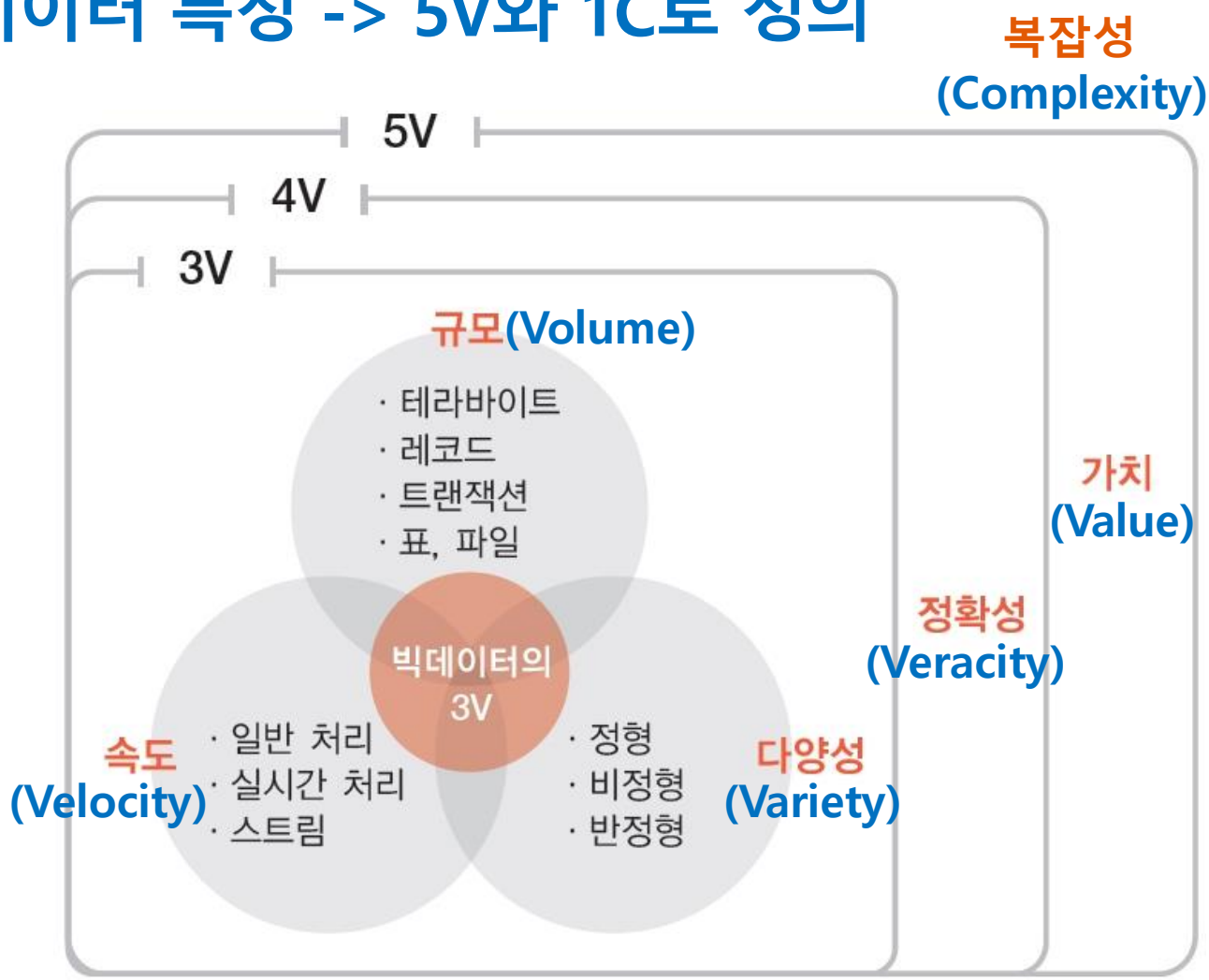
빅데이터 특징



규모 측면	- 전수분석을 통해 정보 왜곡이 줄어듦 - 막대한 데이터가 필요한 새로운 분석 기법 도입 가능
다양성 측면	- 다양한 변수 간의 새로운 관계 발견 - 고객의 형태가 여과 없이 담겨 있는 생생한 비정형 데이터가 핵심
속도 측면	- 사건 발생 시점과 데이터 감지 시점 간의 지연이 거의 없어 실시간 '나우캐스팅 (nowcasting)' 가능

[출처] 빅 데이터: 산업 지각변동의 지원 (삼성경제연구소, 2012년 5월)

빅데이터 특징 -> 5V와 1C로 정의



빅데이터 특징 - 규모(Volumn)

- 처리해야 할 데이터의 크기를 의미하는 속성
- 테라바이트(Terabyte, TB)급 이상의 데이터군을 빅데이터로 통칭
- 정보통신 기술의 발달로 최근에는 더 큰 규모의 데이터를 접할 수 있음

빅데이터 특징 - 다양성(Variety)

- **처리해야 할 데이터의 유형이 다양함을 의미하는 속성**
- **빅데이터는 다양한 데이터 유형**

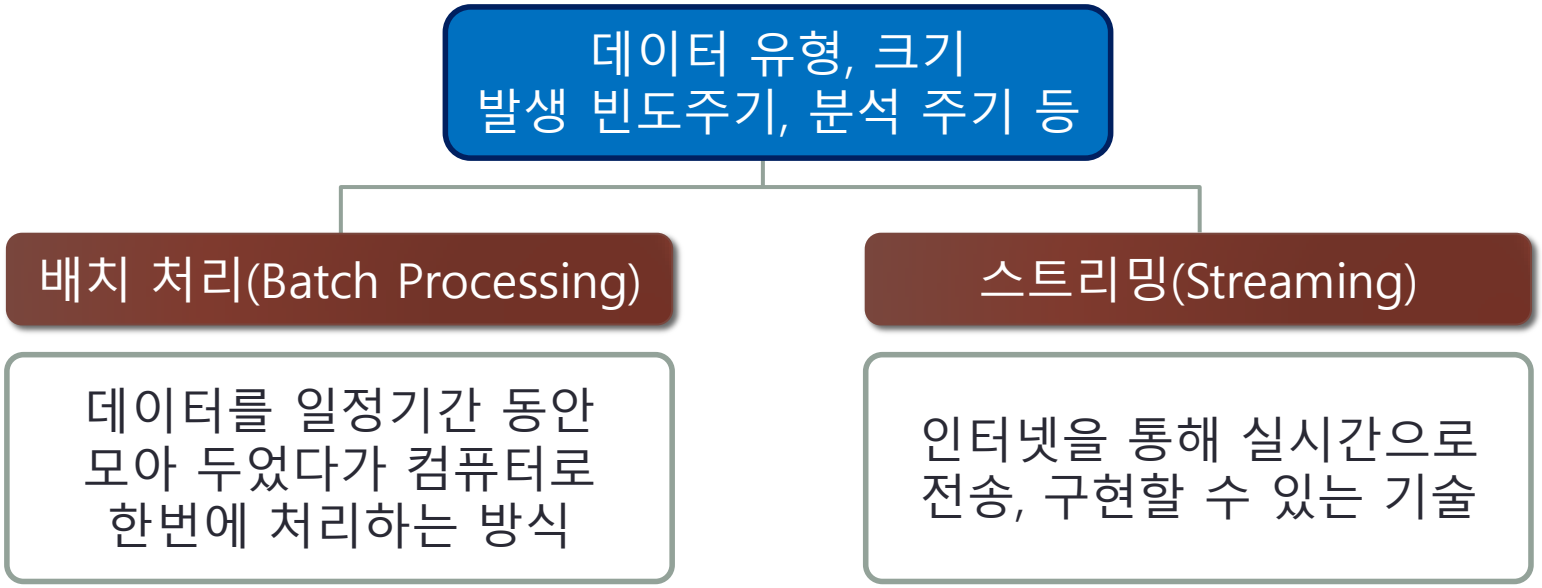
정형 데이터 (Structured Data)	반정형 데이터 (Semi-Structured Data)	비정형 데이터 (Unstructured Data)
데이터 타입이 고정된 필드에 저장된 데이터 관계형 DB, 스프레드시트 등	고정된 필드에 저장되어 있지는 않지만 메타데이터나 스키마를 포함하고 있는 데이터 XML, HTML, 웹 로그, 센서 데이터 등	고정된 필드에 저장되어 있지 않고 형태와 구조가 복잡한 데이터 소셜 데이터, 텍스트 분석이 가능한 문서, 이미지, 음성, 동영상

- **데이터의 양과 더불어 유형의 복잡성이 증가**

  정보통신 기술의 발달로 소셜 데이터, 센서 데이터 등과 같은 관측할 수 있는 데이터가 의미 있는 분석의 대상으로 대두됨

빅데이터 특징 - 속도(Velocity)

- 대용량의 데이터를 빠르게 처리하고 분석할 수 있는 속성
- 데이터의 빠른 처리 및 분석을 위해 다양한 방식을 사용



빅데이터 의미

- 다양하고 방대한 규모의 데이터는 미래 경쟁력의 우위를 좌우하는 중요한 자원으로서 활용 가치가 높음
- 현재의 빅데이터 환경은 과거와 비교해 데이터의 양은 물론 질과 다양성 측면에서 패러다임의 전환을 의미
- 산업혁명 시기의 석탄과 같이 IT와 스마트혁명 시기에 혁신과 경쟁력 강화, 생산성 향상을 위한 아주 중요한 원천으로 간주됨

빅데이터의 환경적 특징

- '디지털 미래 전략(Designing a Digital Future)' 보고서에서 ‘ 모든 연방정부 기관은 빅데이터 전략이 필요함'을 강조(미대통령 과학자문위원회, 2010년 발간)
- 위기에 처한 자본주의를 구하기 위한 '사회 기술 모델(Social and Technological Models)'을 제시하고 '빅데이터'가 사회현안 해결을 위한 강력한 도구가 될 것으로 예측(2012년 다보스 포럼, Vital Wave Consulting, 2012)
- 2011년 '빅데이터를 활용한 스마트 정부 구현(안)'을 보고(대한민국 국가정보화 전략위원회)

빅데이터의 환경적 특징

구분	기존 환경	빅데이터 환경
데이터	- 정형화된 수치자료 중심	- 비정형의 다양한 데이터 - 문자 데이터(SMS, 검색어) - 영상 데이터(CCTV, 동영상) - 위치 데이터
하드웨어	- 고가의 저장장치 - 데이터베이스 - 데이터웨어하우스 (Data-warehouse)	- 클라우드 컴퓨팅과 같은 비용, 효율적인 장비 활용 가능
소프트웨어 분석 방법	- 관계형 데이터베이스(RDBMS) - 통계패키지(SAS, SPSS) - 데이터 마이닝(data mining) - machine learning - knowledge discovery	- 오픈소스 형태의 무료 소프트웨어 - Hadoop, NoSQL - 오픈 소스 통계솔루션(R) - 텍스트 마이닝(text mining) - 온라인 버즈 분석(opinion mining) - 감성 분석(sentiment analysis)

빅데이터의 사회, 경제적 가치

구분	기관명	주요내용
산업 경제성	Economist (2012)	- 데이터는 자본이나 노동력과 거의 동등한 레벨의 경제적 투입 자본, 비즈니스의 새로운 원자재 역할
	Gartner (2011)	- 데이터는 21세기 원유, 데이터가 미래 경쟁 우위를 좌우 - 기업은 다가올 '데이터 경제 시대'를 이해하고 정보 고립(Information Silo)을 경계해야 성공 가능
	Mckinsey (2011)	- 빅데이터는 혁신, 경쟁력, 생산성의 핵심 요소 - 의료, 공공행정 등 5대 분야에서 6천억 불 이상의 가치 창출
국가 경쟁력	미 대통령 과학기술자문위	- 미국 정부기관들이 데이터를 지식으로, 지식을 행동으로 변환하는 전략에 집중해야 함을 주장
	싱가포르	- 데이터를 지반으로 싱가포르를 위협하는 리스크에 대한 평가와 환경변화를 탐지

김현곤, 빅데이터 시대 전망과 대응전략, 한국정보화진흥원, 2012

빅데이터 활용 – 미국 공공부분

모든 연방정부 기관에
빅데이터 전략이 필요하다.

오바마 대통령
과학기술자문위원회(2010)

2012년 대통령 과학기술정책실에서 6개 주요 연방정부가 협력

빅데이터 관련 R&D에 2억 달러 투입하기로 결정

빅데이터 활용 – 미국 공공부분

연방수사국(FBI)	• DNA 색인 시스템 : DNA데이터를 활용해 단시간에 범인을 검거하는 시스템을 운영
국립보건원(NIH)	• 필박스(Pillbox) 프로젝트 : 의약품 정보 서비스를 제공하고 제조사와 사용자 간 유기적으로 정보를 공유하여 후천성면역결핍증 등 관리 대상 주요 질병의 분포와 증감 현황 데이터를 수집하고 분석
미시간 주정부	• 데이터 통합을 통해 공공의료보험(Medicaid) 부정행위 발생 감지, 개인 건강관리 개선, 최적의 입양가정 선택 등 공공 서비스 품질 개선에 활용
오하이오주와 오클라호마주정부	• 국세청(IRS) 데이터와 고용데이터를 분석해 새로운 세원을 확보하고 미납세금을 확인하여 추징하는데 활용

빅데이터 활용 – 기타 국가 공공부분

싱가포르	• 재난방재와 테러감지, 전염병 확산과 같은 불확실한 미래를 대비하기 위해 2004년부터 국가위험관리시스템(RAHS, Risk Assessment & Horizon Scanning)을 추진
영국 국영 보건복지(NHS)	• 국민 건강을 위한 질병 예측 전국의 약국, 병원의 약 처방 데이터를 데이터베이스화 하여 특정 지역 및 특정 질병의 가능성을 분석
영국 패치베이 (Pachube)	• 전력 및 환경 등의 센서 데이터를 개방하고 공유하는 플랫폼 제공 웹 프로그램과 스마트폰 앱 개발등 다양한 사업에 활용

빅데이터 활용 – 국내 공공부분

빅데이터를 활용하여 지식과 정보를 개방하고 상호 협력할 수 있는 스마트한 정부를 구현한다.

- 빅데이터 국가전략 포럼을 통해 정부 및 공공기관, 민간 전문기업, 연구기관, 빅데이터 보유기관과 전문가간의 협력을 활성화 함
- 공공부문의 빅데이터 활용의 성공사례를 조기 발굴하고자 노력함
- 공공 및 민간 데이터의 연계를 활용하도록 유도하고 있음
- 범정부적 빅데이터 전략 로드맵을 수립하고 빅데이터 분석 전문인력 양성에 노력함

빅데이터 활용 – 국내 공공부분

싱가포르	• 재난방재와 테러감지, 전염병 확산과 같은 불확실한 미래를 대비하기 위해 2004년부터 국가위험관리시스템(RAHS, Risk Assessment & Horizon Scanning)을 추진
영국 국영 보건복지(NHS)	• 국민 건강을 위한 질병 예측 전국의 약국, 병원의 약 처방 데이터를 데이터베이스화 하여 특정 지역 및 특정 질병의 가능성을 분석
영국 패치베이 (Pachube)	• 전력 및 환경 등의 센서 데이터를 개방하고 공유하는 플랫폼 제공 웹 프로그램과 스마트폰 앱 개발등 다양한 사업에 활용

빅데이터 활용 – 국내 공공부분

국민연금공단	• 국민연금 가입자의 의견을 분석해 불신을 해소하고 소통하기 위해 여론정보 수집분석시스템 운영
국민권익위원회	• 국민의 의견을 분석해 불신을 해소하고 소통하기 위해 민원동향분석시스템 운영
국가정보화 전략위원회	• 공공 부문의 빅데이터 활용 시나리오를 재난 전조 감지, 구제역 예방, 사회복지 통합 관리망 구축으로 맞춤형 복지 서비스 제공, 물가 관리, DNA, 의료데이터 공유와 활용 촉진을 통해 개인맞춤형 의료 시스템 구축의 다섯 분야로 제시

빅데이터 활용 – 기업의 활용

- **구글 자동번역 시스템 :**
  통계적 기계 번역(statistical machine translation)라고 표현
- **IBM연구소의 슈퍼컴 '왓슨' :**
  퀴즈쇼〈제퍼디(Jeopardy!)〉에 출연해 인간 챔피언과 겨뤄 승리
- **아마존(Amazon) :**
  고객의 도서 구매 데이터를 분석해 특정 책을 구매한 사람이 추가 구매할 것으로 예상되는 도서 추천 시스템 개발
- **일본의 최대 전자상거래 업체인 라쿠텐(樂天) :**
  슈퍼 데이터베이스(DB)를 구축해 이를 기반으로 그룹 내 전자상거래 사업과 신용·결제 서비스, 포털, 여행, 증권, 프로 스포츠사업 부문에서 공동 활용하고 있음

빅데이터 활용 – 기업의 활용

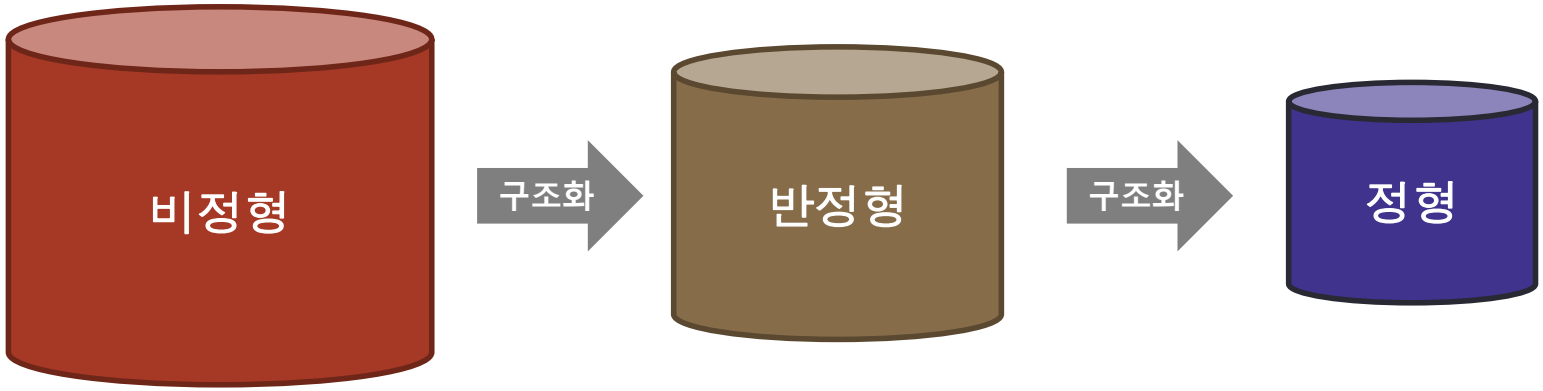
- **미디어 콘텐츠 유통기업인 넷플릭스(Netflix) :**
  이용자 영화대여 목록을 기초로 새로운 영화를 추천하는 시네매치(Cinematch) 시스템 개발
- **패스트 패션(fast fashion)의 선도자인 자라(Zara) :**
  빅데이터 활용을 위해 MIT 연구팀과 최적 재고관리 시스템을 개발하여 현재 유행하는 패션 트렌드를 즉시 반영해 단기간에 다품종 소량 생산하는 초스피드 전략을 채택, 이러한 전략을 뒷받침하기 위해서 상품 수요를 예측하고 매장별 적정 재고를 산출하여 상품별 가격 결정과 운송 계획까지 실시간 데이터 분석에 의존하고 있음

빅데이터 처리

그림 1-7 빅데이터 처리 과정 [09]

데이터 종류

종류	설명
정형	고정된 필드에 저장된 데이터 예 관계형 데이터베이스, 스프레드시트
반정형	고정된 필드에 저장되어 있지는 않지만, 메타데이터나 스키마 등을 포함하는 데이터 예 XML, HTML 텍스트
비정형	고정된 필드에 저장되어 있지 않은 데이터 예 텍스트 분석이 가능한 텍스트 문서, 이미지 · 동영상 · 음성 데이터

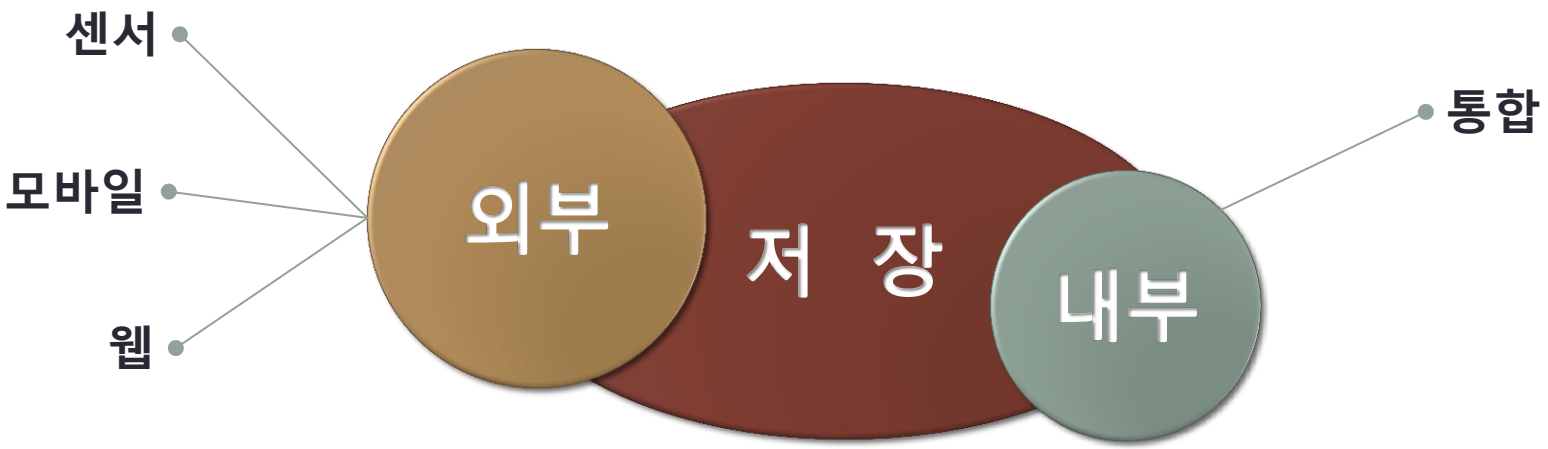


데이터 처리 변화

데이터의 저장,관리,분석의 전체 과정을 빅데이터에 적용하려면 데이터 처리 방식은 달라져야 한다.

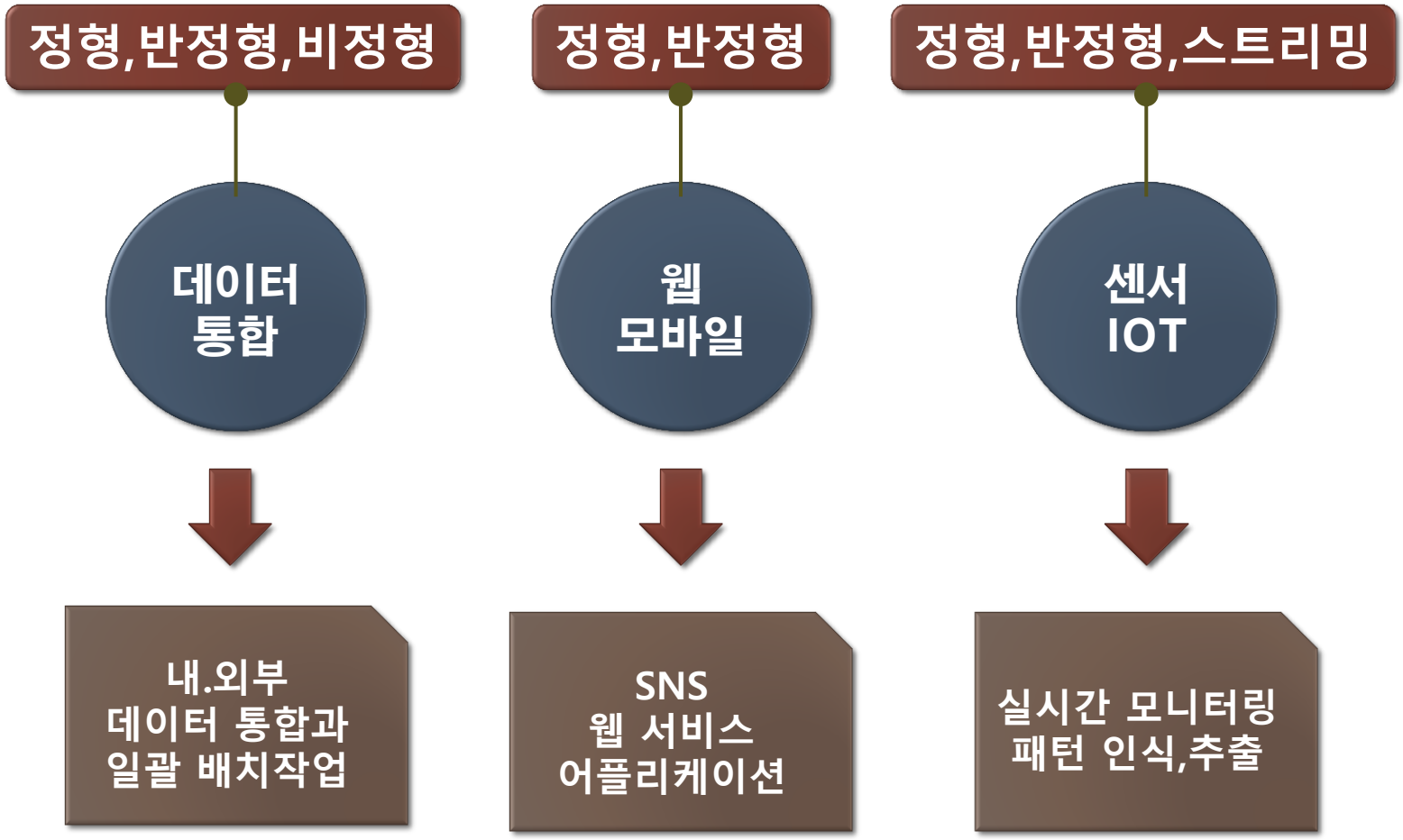
구분	정보화 시대	스마트 시대
저장	관계형(정형) 데이터베이스, 데이터웨어하우스	비관계형(비정형) 데이터베이스, 가상화, 클라우드 서비스 등
관리	정보.지식 관리 시스템, 웹 기반 2.0	플랫폼, 소셜 네트워크, 집단지정
분석	경영 정보, 고객 정보, 자산 정보 분석 등 (ERP, CRM, 데이터마이닝 등)	빅데이터 고급 분석 (소셜 분석, 다각화, 시각화)

원천 소스



외부데이터	센서
	웹, 모바일
내부 데이터	통합 ETL

처리 및 서비스


정형,반정형,비정형
데이터 통합
내.외부 데이터 통합과 일괄 배치작업
정형,반정형
웹 모바일
SNS 웹 서비스 어플리케이션
정형,반정형,스트리밍
센서 IOT
실시간 모니터링 패턴 인식,추출

구글의 빅데이터 처리 기술

- 구글의 검색엔진 기술은 크게 3가지로 요약 할 수 있다.

- 대량의 정보를 효과적으로 저장하기 위한 분산파일 시스템 (GFS, Google File System),
- 대용량 데이터의 읽기와 쓰기를 위한 분산 스토리지 시스템 빅테이블(Bigtable)
- 분산 데이터 처리를 위한 맵리듀스(MapReduce)

구글의 빅데이터 처리 기술 – 분산 파일 시스템(GFS)

- 구글의 분산파일시스템(GFS)은 여러 대의 컴퓨터를 조합해 대규모 기억장치(storage)를 만드는 기술
- 웹 검색엔진의 경우 전 세계에 존재하는 엄청난 규모의 웹 페이지 를 저장해야 한다. 인터넷 상의 데이터는 그 증가 속도가 매우 빠르기 때문에 대규모 데이터를 안전하게 저장하고 효율적으로 처리 하기 위해서는 다수의 하드디스크를 조합해 데이터를 저장하는 새로운 기술이 필요하다.(니시다 케이스케, 2009).
  분산파일 시스템은 이를 위해 개발된 구글의 독자적인 기술
- 구글은 가격이 저렴한 하드웨어를 대량으로 이용하기 때문에 고장 발생을 전제로 시스템 설계.
- 분산파일 시스템은 이를 위해 항상 파일과 파일의 내용과 위치에 대한 정보도 여러 개 복사해 저장한다. 또한 여러 개의 복사본을 만들어 저장 이렇게 파일의 내용과 정보가 여러 대의 컴퓨터에 분 산 저장되기 때문에 검색 시간도 단축되고 여러 곳에서 동시에 검색이 이루어져도 어느 한 곳에 작업량이 집중되지 않고 한 대의 컴퓨터가 고장이 나도 거기에 담겨 있는 정보는 다른 곳에 복사본 이 존재하기 때문에 데이터 손실의 염려도 없다.

구글의 빅데이터 처리 기술 – 빅테이블(Big Table)

- 구조화된 데이터(Structured Data) 처리를 위한 분산스토리지시스템 (A Distributed Storage System, Fay Chang, 2006).
- 웹 검색과 같은 대규모의 복잡한 데이터 구조에서 효율적으로 읽 고 쓰기 위해 빅테이블은 기존의 관계형 데이터베이스와 달리 복잡한 구조를 가지고 있다. 관계형 데이터베이스가 테이블(Table), 로우(Row), 컬럼(column)이라는 간단한 구조로 구성되어 있는 반면 빅테이블은 컬럼 대신에 로우 키(Row Key)와 컬럼 패밀리 (column family), 타임 스탬프(time stamp)와 같은 복잡한 구조 로 구성되어 있다. 빅테이블은 이러한 기능을 이용해 테이블을 종횡으로 무한정 늘려 갈 수 있다.(Fay Chang, 2006)

구글의 빅데이터 처리 기술 – 맵리듀스(MapReduce)

- 효율적인 데이터 처리를 위해 여러 대의 컴퓨터를 활용하는 분산 데이터 처리기술(Dean&Ghemawat, 2004).
- 맵(Map)과 리듀스(Reduce)의 두 과정으로 구성.
  먼저 맵 단계에 서는 대규모 데이터를 여러 대의 컴퓨터에 분산해 병렬적으로 처 리해 새로운 데이터(중간 결과)를 만들어낸다. 리듀스 단계에서는 이렇게 생성된 중간 결과물을 결합해 최종적으로 원하는 결과를 생산한다. 리듀스 과정 역시 여러 대의 컴퓨터를 동시에 활용하는 분산처리 방식을 적용

오픈소스 하둡 프로젝트

- 데이터 분산 저장
- 네트워크 최대 활용
- 효율적인 백업 • 복구
- 분산 환경 데이터 처리
- 연관 데이터 조합
- 분산 저장 활용

- 오픈소스 하둡의 핵심 기능은 **분산저장과 분산병렬처리**이다.
- 오픈소스 하둡은 저장과 처리에 대한 기본적인 기능만 제공하기 때문에 데이터를 수집하고 분석하기에는 부족하다.
- 하둡을 보완하기 위한 **하둡 기반의 S/W들을 에코시스템**이라 한다.

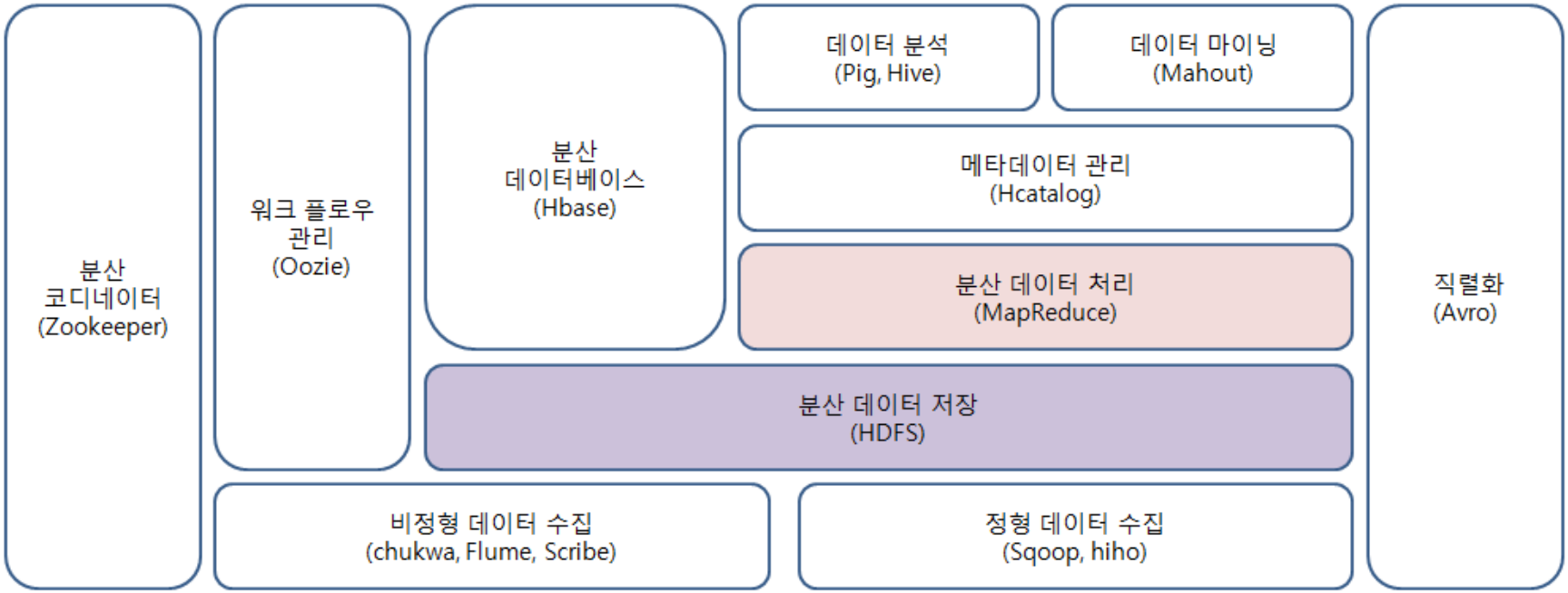
하둡의 특징

- 신뢰성 있고, 확장성 있는 대용량 데이터 처리를 위해 개발된 분산 컴퓨팅을 위한 오픈소스 프레임워크(open-source framework)
- 2005년 당시 오픈소스 검색엔진 “Nutch”를 개발 중이던 더그 커팅이 대용량 데이터 처리를 위해 2004년 발표된 구글의 논문을 참조하여 구현
- 야후 (Yahoo)의 재정지원으로 2006년부터 본격적으로 개발되었으며 현재는 아파치(Apache) 프로젝트로 분리되어 개발되고 있음
- 구글의 분산 파일 시스템(GFS)는 하둡 분산파일 시스템(HDFS, Hadoop Distributed File System), 구글의 분산 병렬 처리 시스템(MapReduce)은 하둡의 맵리듀스(Hadoop MapReduce), 구글의 분산 데이터 베이스인 빅테이블은 에이치베이스(Hbase)를 사용해 구현
- 하둡이란 명칭은 하둡 개발자인 더그 커팅(Doug Cutting)이 자신의 아이가 가지고 놀던 장난감 코끼리의 이름을 붙인 것에서 기인

하둡의 특징

- GFS, MapReduce 소프트웨어 구현체
  - 아파치 Top-Level 프로젝트
  - 하둡 코어는 Java, Python, C/C++ 등을 지원
- 대용량 데이터 처리를 위한 플랫폼
  - 분산 파일 시스템(HDFS)
  - 분산 병렬 처리 시스템(MapReduce)
  - 기반 소프트웨어 프레임워크(Core)

하둡 에코시스템 구조


분산
코디네이터
(Zookeeper)
워크 플로우
관리
(Oozie)
분산
데이터베이스
(Hbase)
데이터 분석
(Pig, Hive)
데이터 마이닝
(Mahout)
메타데이터 관리
(Hcatalog)
분산 데이터 처리
(MapReduce)
분산 데이터 저장
(HDFS)
직렬화
(Avro)
비정형 데이터 수집
(chukwa, Flume, Scribe)
정형 데이터 수집
(Sqoop, hiho)

Chapter 02

Hadoop 2.0 소개

- 하둡 history
- 하둡 2.0의 구성
- 하둡 디렉터리 구성
- 에코시스템 구성

하둡 history

- 2002 : 웹 검색엔진 너치프로젝트(Apache Nutch)가 개기 .
- 2003 : Google의 GFS (Google File System)
- 2004 : Google의 대규모 분산 프로그래밍 모델 'MapReduce'
- 2004 : 더그커팅이 검색엔진 오픈소스 개발 (Building Nutch)
  (너치 분산파일 시스템 + 맵리듀스)
- 2005 : Doug Cutting에 의해 Hadoop 오픈소스 개발 .
- (HDFS+MapReduce)
- 2006 : Nutch project와 yahoo가 병합.
- 2008 : yahoo는 10,000개의 하둡 코어를 이용한 야후 서비스 발표.

하둡 2.0의 구성

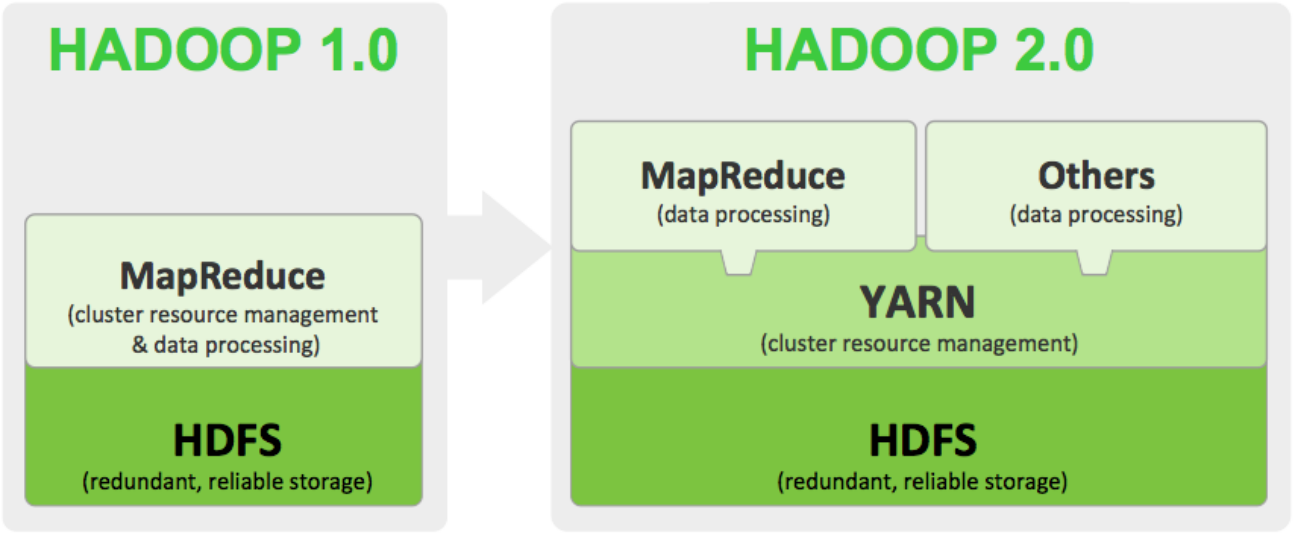
- **Hadoop Common**
  - 다른 하둡 모듈을 지원하는 유틸리티
- **HDFS(Hadoop Distributed File System)**
  - 애플리케이션 데이터에 고성능 접근을 지원하기 위한 분산 파일 시스템
- **Hadoop YARN(Yet Another Resource Negotiator)**
  - 잡 스케줄링과 클러스터 리소스를 관리하기 위한 모듈
- **Hadoop MapReduce**
  - 대용량 데이터의 병렬 처리를 위한 얀 기반 시스템
- **Hbase**
  - 컬럼 기반의 데이터베이스(column-oriented database)로 대규모 데이터에 빠른 속도로 접근할 수 있도록 하는 분산 데이터 베이스

하둡 2.0의 구성

- 하둡의 핵심 구성 요소인 HDFS와 맵리듀스 이외에 하둡 프로그램을 쉽게 처리 하기 위한 솔루션으로 피그(Pig)와 하이브(Hive)가 있음
- 피그
  - 데이터를 적재·변환, 결과를 정렬하는 과정을 쉽게 처리하기 위해 만든 프로그램 언어
- 하이브
  - 하둡을 데이터웨어하우스(DW)로 운영할 수 있게 해주는 솔루션
- 스쿱(Sqoop)
  - 관계형 데이터베이스로 데이터를 하둡으로 옮기는 도구
- 플럼(Flume)
  - 로그데이터를 하둡 분산파일 시스템으로 옮기는 도구
- 이 밖에 처리 과정을 조정하고 관리하는 주키퍼(Zookeeper)와 우지(Oozie)가 있음

하둡 2.0의 구성

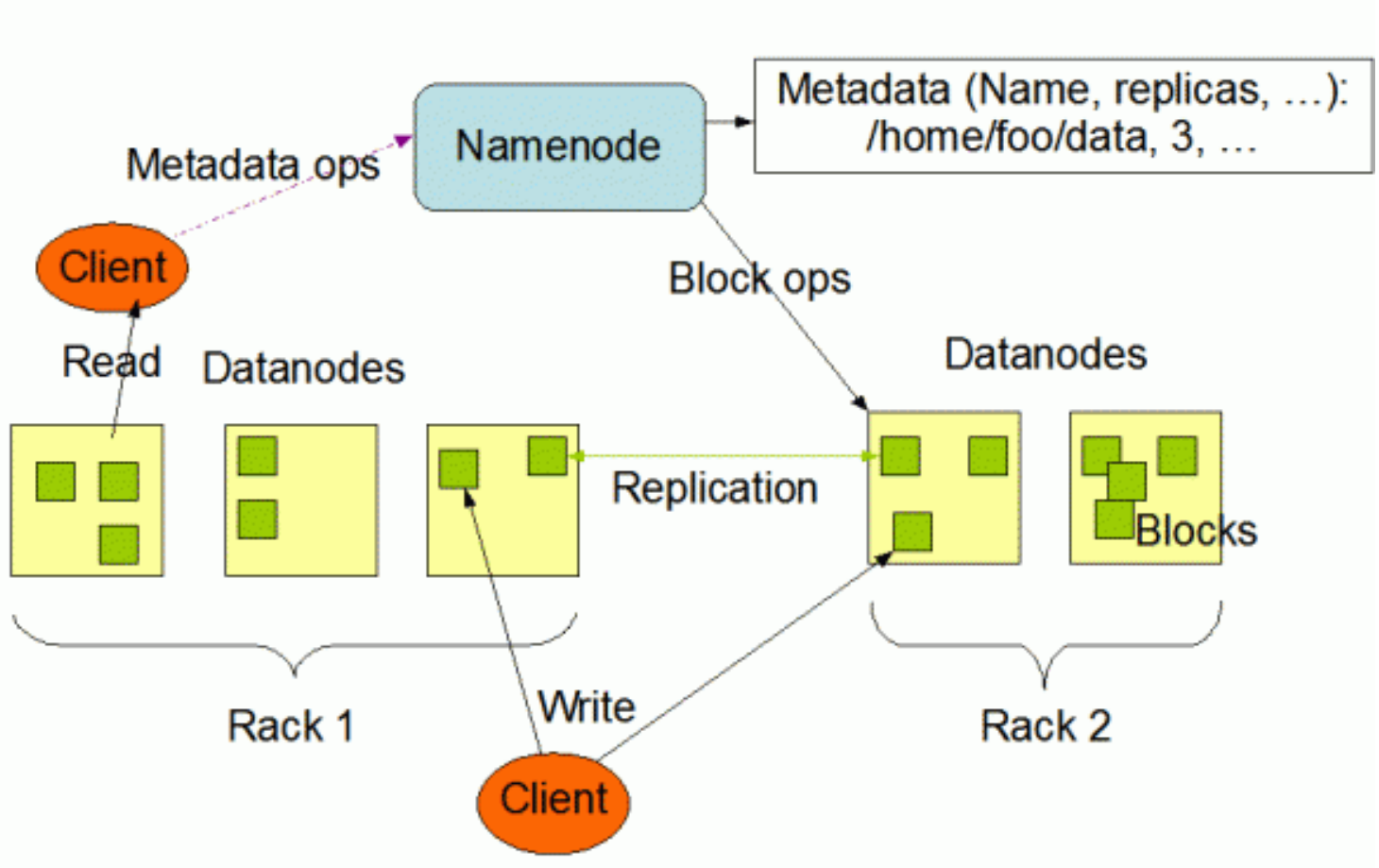
- Hadoop 1.0의 단점
  - 한 노드에서 실행할 수 있는 Map과 Reduce용 작업숫자가 제한되어, 노드에 여유 자원이 있어도 그 자원을 활용 하지 못하는 상황이 발생. (자원 분배 및 작업 관리의 비효율성)
- YARN (Yet Another Resource Negotiator)
  - 자원 관리, Job 상태 관리를 ResourceManager와 ApplicationMaster로 분리하여, 기존 Job Tracker에 몰리던 병목을 제거
  - MapReduce 외에 다양한 어플리케이션을 실행할 수 있으 며, 어플리케이션마다 자원(CPU, 메모리)을 할당 받음



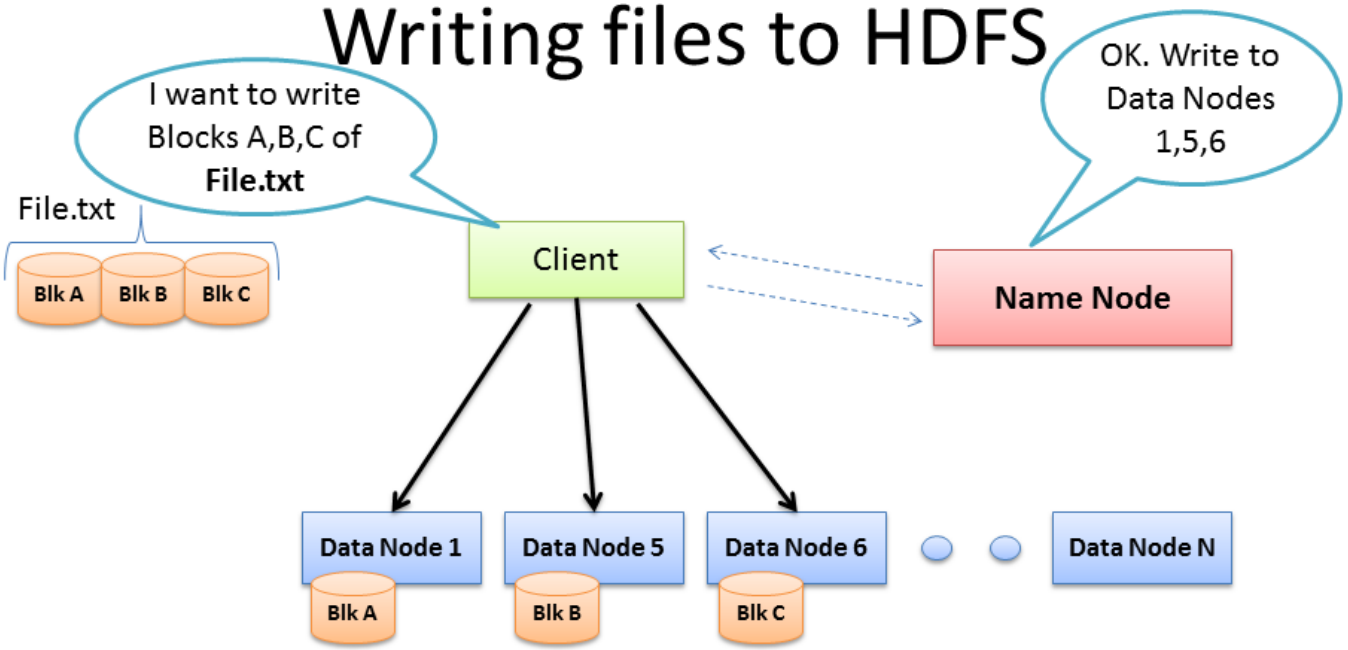
HDFS(Hadoop Distribute File System)

- 파일의 분산 저장이 목적
- NameNodes와 DataNodes로 구성
  - Master NameNode
  - Secondary NameNode
  - DataNode
- 저렴한 컴퓨터로 대 용량 데이터를 저장할 수 있는 시스템
  - 네트워크 Raid와 같이 연결된 것 처럼 사용하는 하드디스크
  - Scale Out
- Block(Chunk) 단위로 파일관리 (저장/복제/삭제)
  - Default Size는 128M(134217728)
- 복제기능을 통해 안전성/신뢰성을 보장
- 1대의 Master서버에 4000+이상의 DataNodes를 운영할 수 있음.
- API(Application Programming Interface)지원
  - 하둡 코어는 Java, Python, C/C++ 등의 프로그래밍 언어를 지원

HDFS 구성


Metadata ops
Namenode
Metadata (Name, replicas, …):
/home/foo/data, 3, …
Client
Block ops
Read
Datanodes
Datanodes
Replication
Blocks
Rack 1
Write
Rack 2
Client

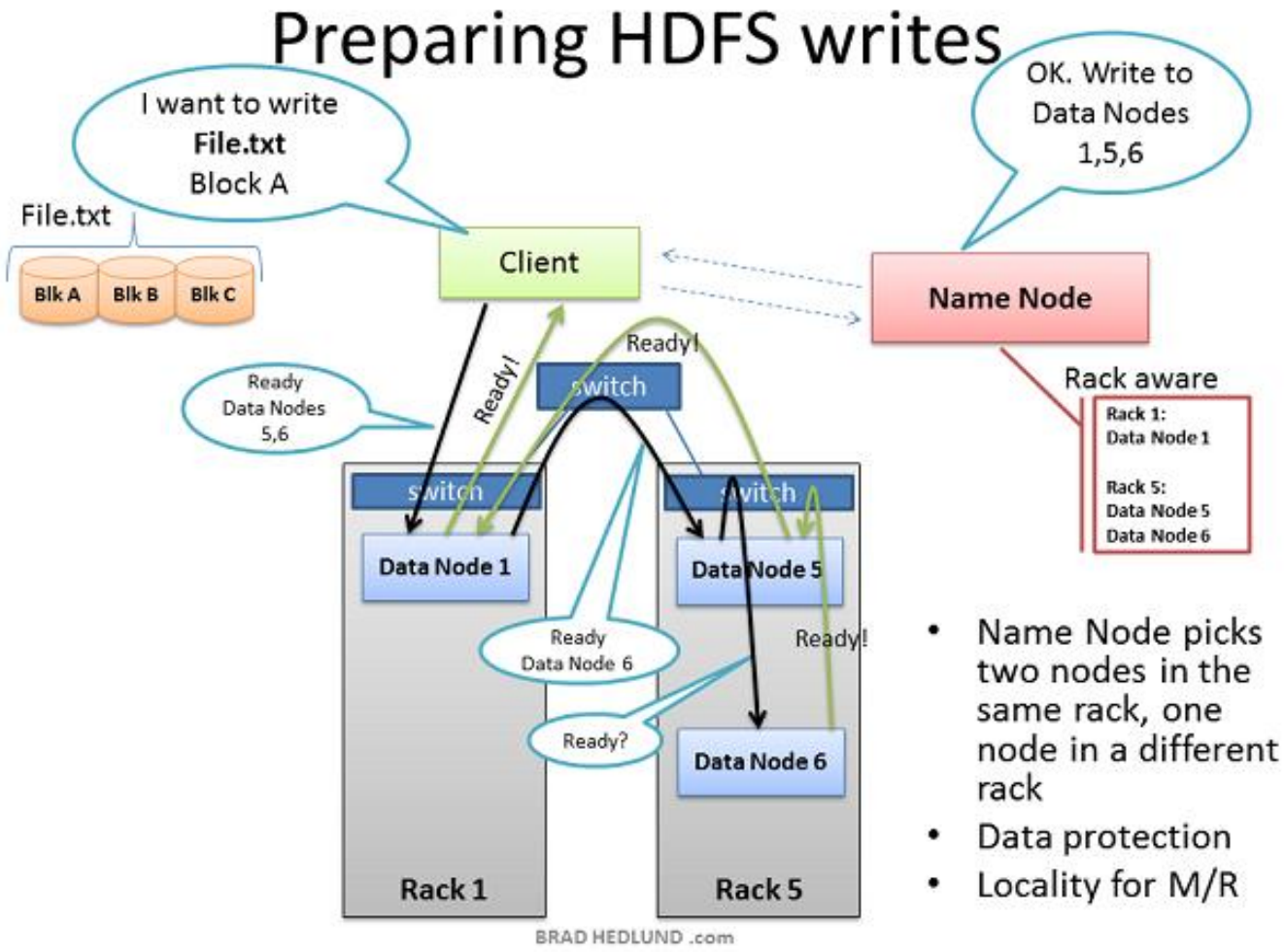
HDFS의 파일쓰기



- Client consults Name Node
- Client writes block directly to one Data Node
- Data Nodes replicates block
- Cycle repeats for next block

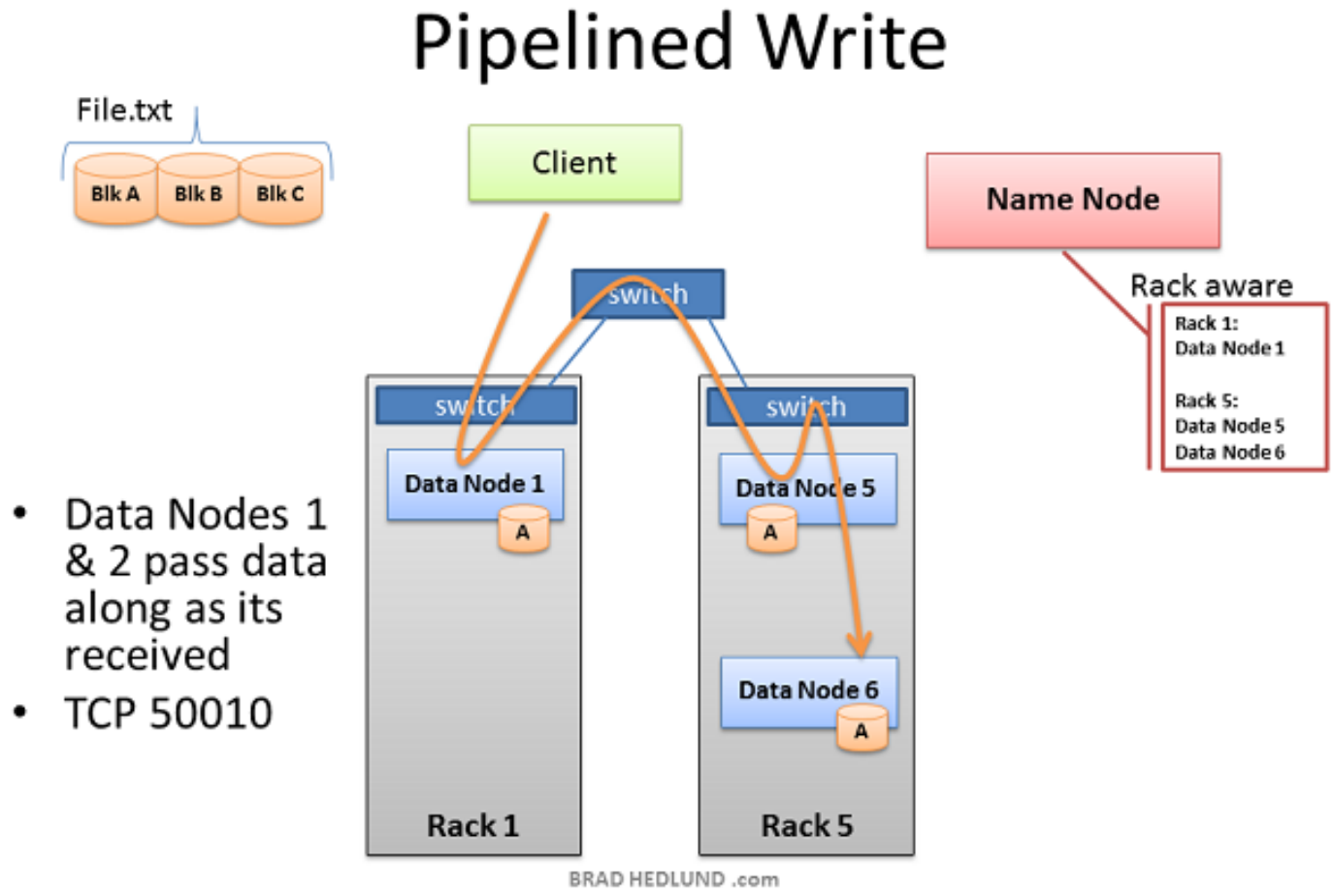
BRAD HEDLUND .com

HDFS의 파일복제(1)



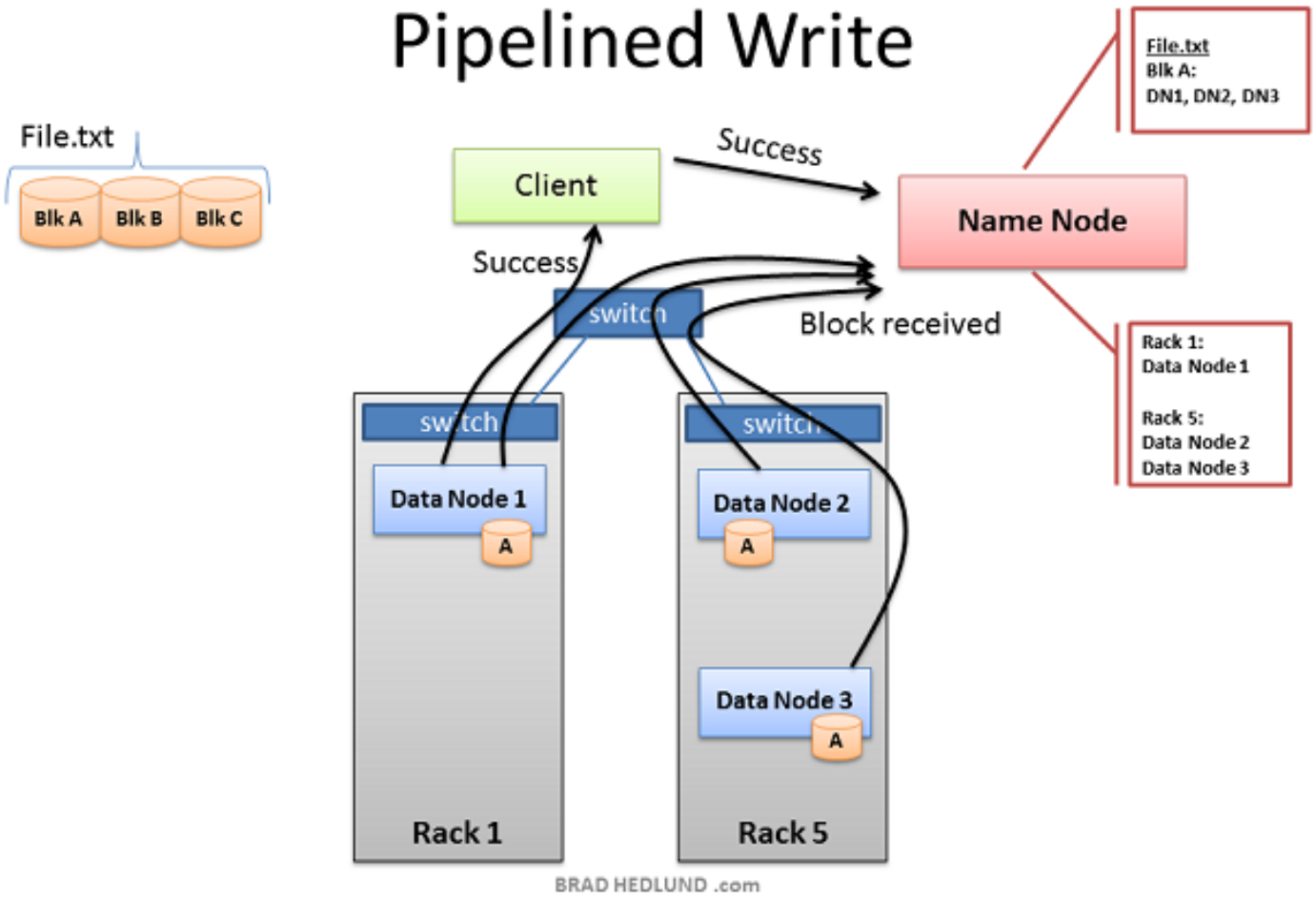
- Name Node picks two nodes in the same rack, one node in a different rack
- Data protection
- Locality for M/R

HDFS의 파일복제(2)

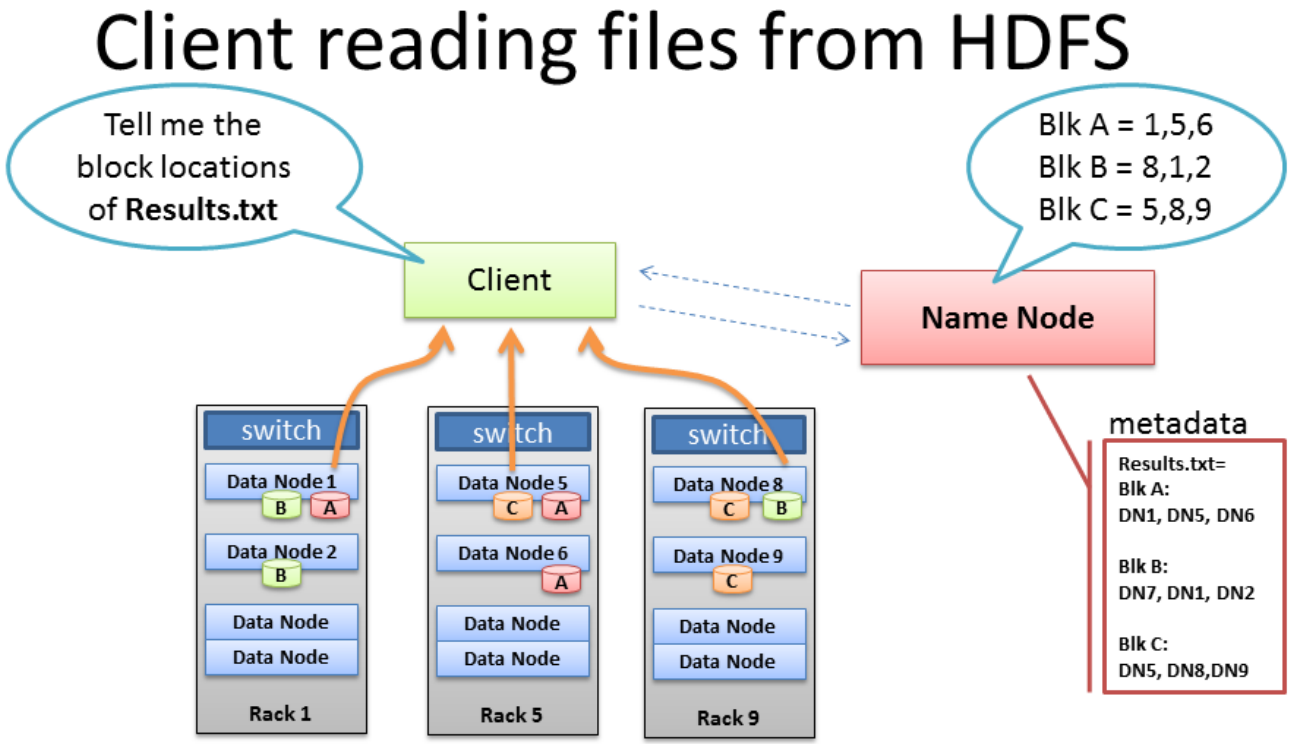


- Data Nodes 1 & 2 pass data along as its received
- TCP 50010

HDFS의 파일복제(3)


Pipelined Write
File.txt
Blk A
Blk B
Blk C
Client
Success
Name Node
File.txt
Blk A:
DN1, DN2, DN3
Success
switch
Block received
Rack 1:
Data Node 1
Rack 5:
Data Node 2
Data Node 3
switch
switch
Data Node 1
A
Data Node 2
A
Data Node 3
A
Rack 1
Rack 5
BRAD HEDLUND .com

HDFS의 파일읽기

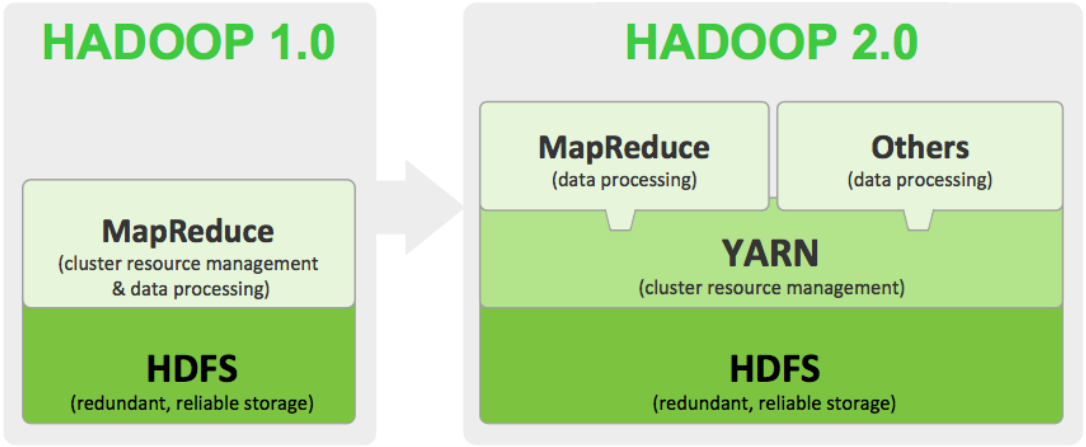


- Client receives Data Node list for each block
- Client picks first Data Node for each block
- Client reads blocks sequentially

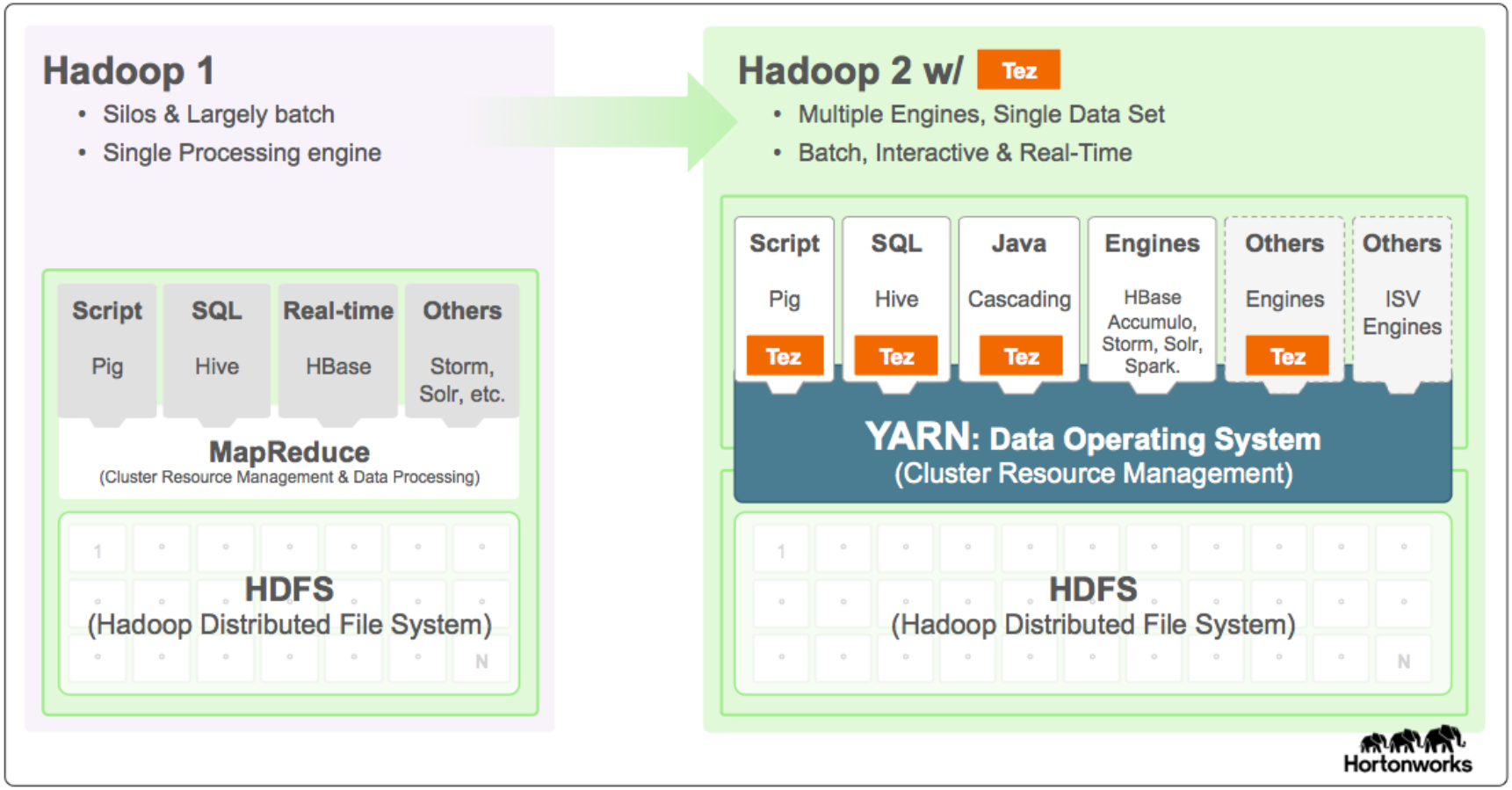
BRAD HEDLUND .com

YARN

- Yet Another Resource Negotiator
- JobTracker의 두 가지 역할 분리
  ① Resource 관리
  ② Job 상태 관리 : 기존 JobTracker의 병목을 제거
- 범용 클러스터 API
  - MapReduce 외에 다양한 어플리케이션을 실행할 수 있으며, 어플리케이션 마다 자원(CPU,메모리)을 할당

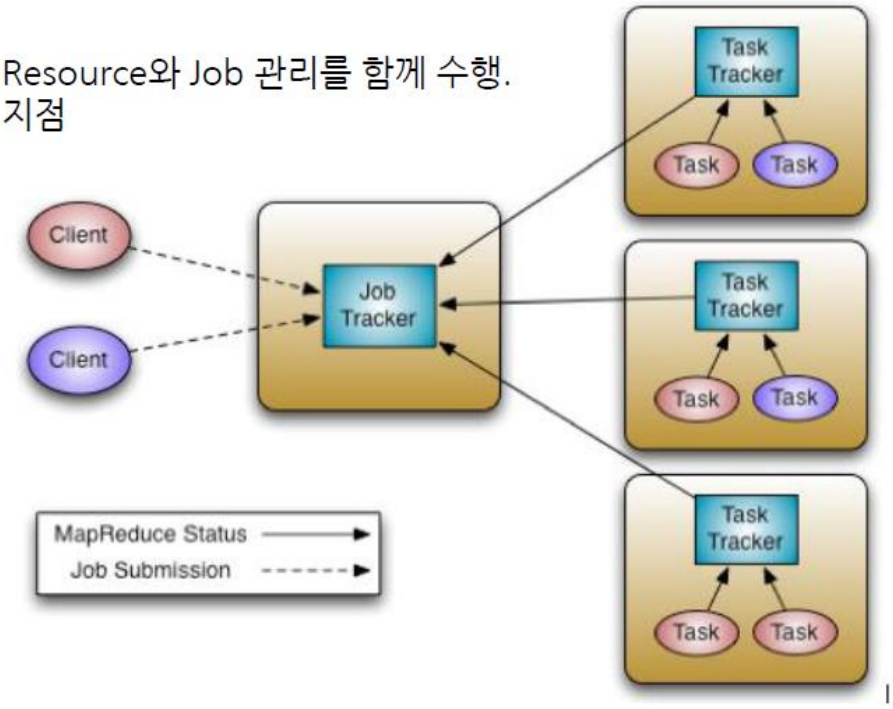


YARN


Hadoop 1
• Silos & Largely batch
• Single Processing engine
Script
Pig
SQL
Hive
Real-time
HBase
Others
Storm, Solr, etc.
MapReduce
(Cluster Resource Management & Data Processing)
HDFS
(Hadoop Distributed File System)
Hadoop 2 w/ Tez
• Multiple Engines, Single Data Set
• Batch, Interactive & Real-Time
Script
Pig
Tez
SQL
Hive
Tez
Java
Cascading
Tez
Engines
HBase Accumulo, Storm, Solr, Spark.
Others
Engines
Tez
Others
ISV Engines
YARN: Data Operating System
(Cluster Resource Management)
HDFS
(Hadoop Distributed File System)
Hortonworks

YARN – Hadoop 1.x

Master Node
하둡 클러스터의 Resource와 Job 관리를 함께 수행.
병목이 발생하는 지점

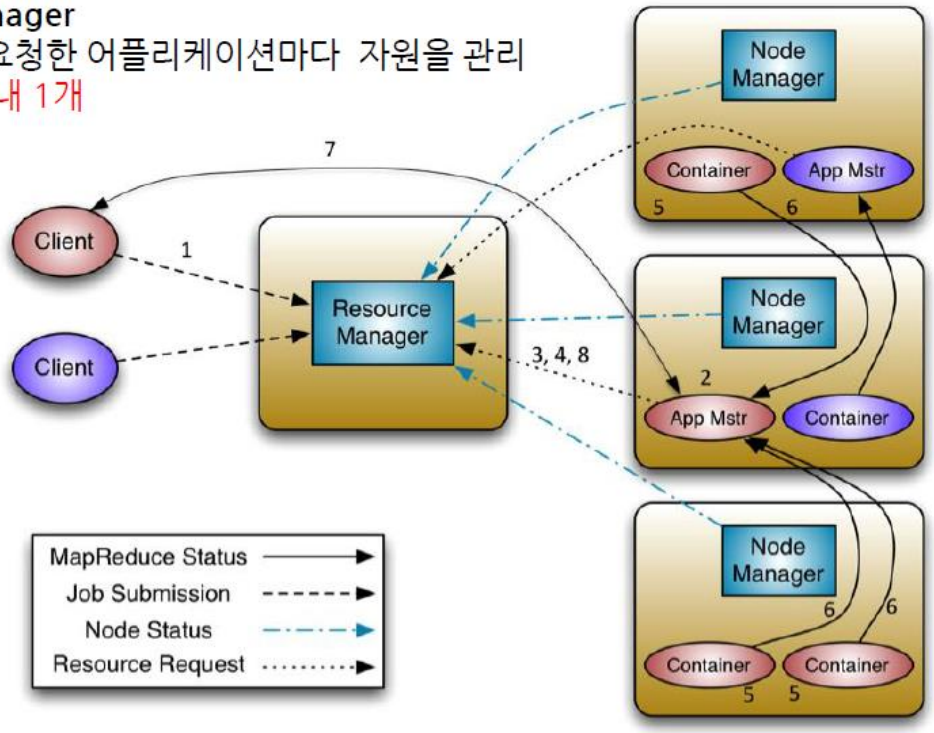


Data Node
한 노드에서 실행할 수 있는
Map과 Reduce Task의 개수 제한.

M/R만 처리

YARN - Hadoop 2.x

Resource Manager
클라이언트가 요청한 어플리케이션마다 자원을 관리
하둡 클러스터 내 1개



Node Manager
각 슬레이브 노드 마다 1개.
컨테이너와 자원의 상태를
RM에게 통지

Application Master
어플리케이션의 실행을 관리하고
상태를 RM에게 통지
어플리케이션마다 1개.

Container
어플리케이션을 수행하는 역할
제한된 자원을 소유하며,
상태를 AM에게 통지

출처 : http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

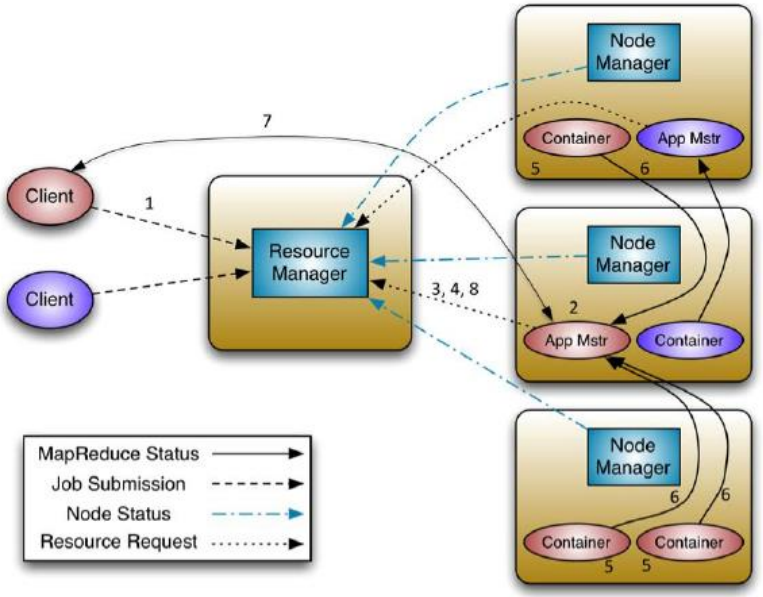
YARN - Hadoop 2.x

1. 클라이언트가 RM에게 어플리케이션 제출
2. NM을 통해 AM 실행
3. AM은 RM에게 자신을 등록
4. AM은 RM에게 컨테이너 할당할 공간/위치를 받음
5. AM은 NM에게 컨테이너를 실행 요청

   (어플리케이션 정보를 NM에게 제공)
6. 컨테이너는 어플리케이션의 상태정보를 AM에 알림
7. 클라이언트는 어플리케이션의 실행정보를 얻기 위해

   AM와 직접 통신
8. 어플리케이션 종료되면 AM은 RM에게서

   자신의 자원을 해제하고 종료



출처 : http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

MapReduce

- 구글에서 분산 컴퓨팅을 지원하기 위한 목적으로 제작해 2004년 발표한 소프트웨어 프레임워크, HDFS에 저장된 파일이용
- 프레임워크는 페타바이트(Petabyte) 이상의 대용량 데이터를 신뢰할 수 없는 컴퓨터로 구성된 클러스터 환경에서 병렬 처리를 지원하기 위한 목적으로 개발됨
- 프레임워크는 함수형 프로그래밍에서 일반적으로 사용되는 map()과 reduce() 함수 기반으로 구성
  - map() : 데이터를 key와 value 한 쌍으로 처리하여 또 다른 key와 value 쌍을 생성하는 함수
  - reduce() : 맵으로 부터 생성된 (key, list(value))들을 병합(merge)하여 최종적으로 list(value)를 생성하는 함수
- Java를 사용하기 때문에 Java 언어를 다를 수 있어야 사용이 가능함

하둡 주요 디렉토리(1)

- **Hadoop home**

이것은 하둡 소프트웨어가 설치되는 디렉토리이다. 이름과 달리 사용자 홈 디렉토리에 설치되지 않는다. 이 디렉토리는 정확하게 설정했다면 읽기 전용이 돼야 하며, 디렉토리 위치는 /usr/local 또는 /opt 이고, 패키지를 통해 설치했다면 /usr 이다.

- **Datanode Data Directory**

HDFS 블록을 저장하기 위해 datanode는 하나 또는 다수의 디렉토리를 사용한다. 데이터노드는 각각의 디렉토리가 분리된 물리 디바이스(독립적인 축과 회전)에 있다고 가정하고 block을 여러 디스크에 순차적으로 저장한다. 이 디렉토리는 디스크 용량이 매우 크고 데이터의 장기 보관용으로 사용된다. 보통 TaskTracker의 Mapreduce local directory와 동일한 디바이스를 사용한다.

- **Namenode directory**

namenode는 File System Meta-data를 저장하기 위해 하나 또는 다수의 디렉토리를 사용한다. namenode는 각 디렉토리가 분리된 물리 디바이스에 있다고 가정하고 디스크 장애시 데이터 가용성을 보장하기 위해 데이터를 모든 디렉토리에 중복 저장한다. 이런 디렉토리들은 모두 동일한 용량의 공간이 필요하고 일반적으로 100GB를 넘지 않는다. 이 디렉토리 중 하나는 일반적으로 NFS이므로 데이터 사본은 물리적인 머신 외에도 보관된다.

하둡 주요 디렉토리(2)

- **Mapreduce local directory**

TaskTracker는 Mapreduce job을 실행할 때 발생하는 임시 데이터를 저장하기 위해 하나 또는 다수의 디렉토리를 이용한다. Mapreduce task들이 서로 간섭을 적게 할 수록 Mapreduce는 잘 돌아가고 성능도 매우 좋아진다. 이 디렉토리는 mapreduce job의 특성에 따라 조금씩 다르기는 하지만 평균적인 용량의 데이터를 저장한다. 보통 datanode의 data directory와 동일한 디바이스를 사용한다.

- **Hadoop log directory**

이것은 job과 task의 임시 데이터는 물론 로그 데이터를 저장하기 위해 모든 데몬이 사용하는 디렉토리이다. 하둡에서 로그 데이터의 크기는 클러스터 사용량과 비례한다. Mapreduce job이 많을 수록 로그도 많아진다.

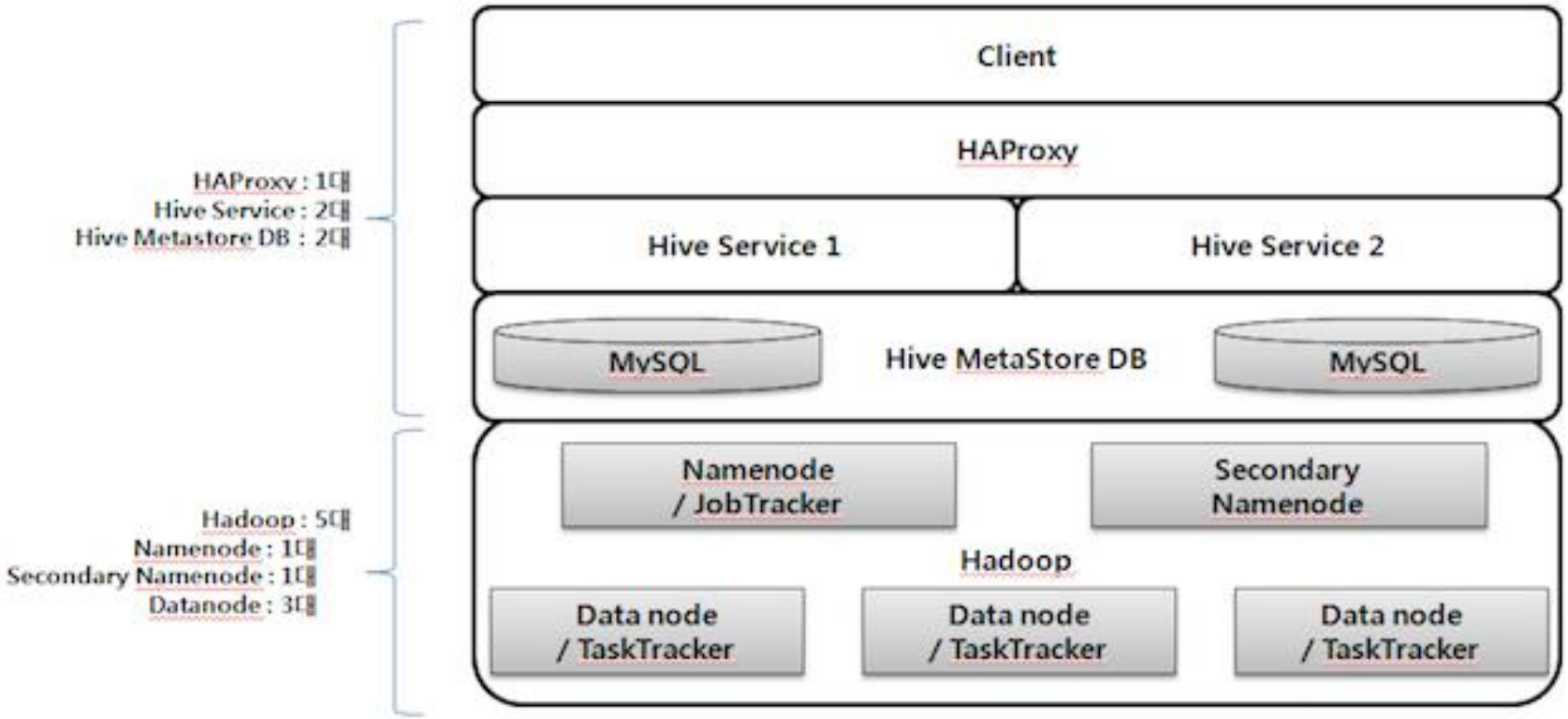
- **Hadoop PID directory**

이것은 PID 파일을 저장하기 위해서 모든 데몬이 사용하는 디렉토리이다. 데이터는 아주 적고 증가하지 않는다.

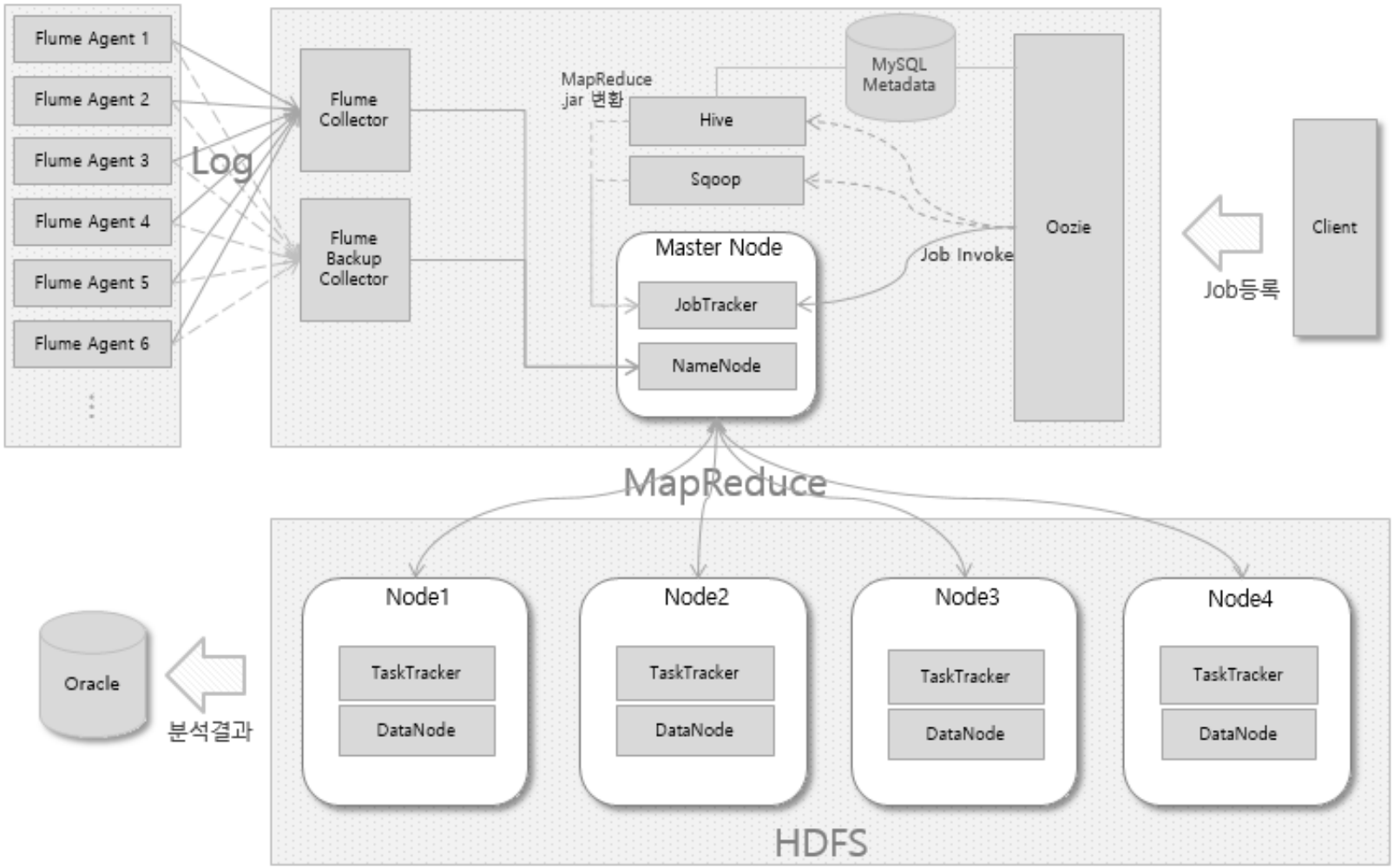
- **Hadoop temporary**

Hadoop은 작은 임시 파일을 저장하기 위해 임시 디렉토리를 이용한다. 임시 디렉토리는 Mapreduce job이 제출되면 이후에 jobtracker가 보내준 JAR 파일 사본을 저장하므로 머신에서 자주 사용된다. 디렉토리 기본위치는 /tmp/hadoop-<${user.name}> 이고 많은 관리자가 그대로 둔다.

Hadoop Cluster Example



Hadoop Ecosystem Example



Chapter 03 Hadoop 클러스터 환경 구축

- 가상 환경 구축
- 리눅스(CentOs) 다운로드
- CentOS 설치
- DataNode 구축

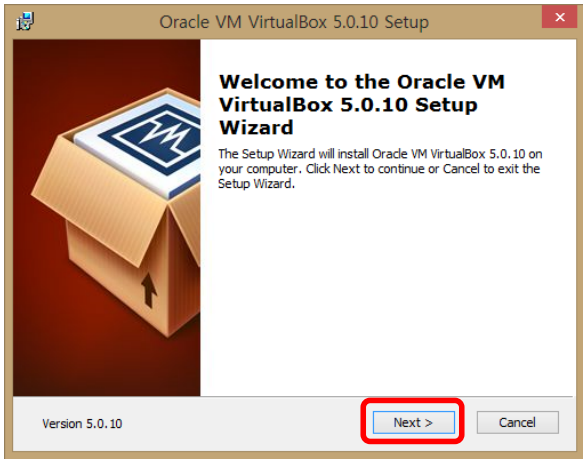
가상 환경 구축(VirtualBox Download)

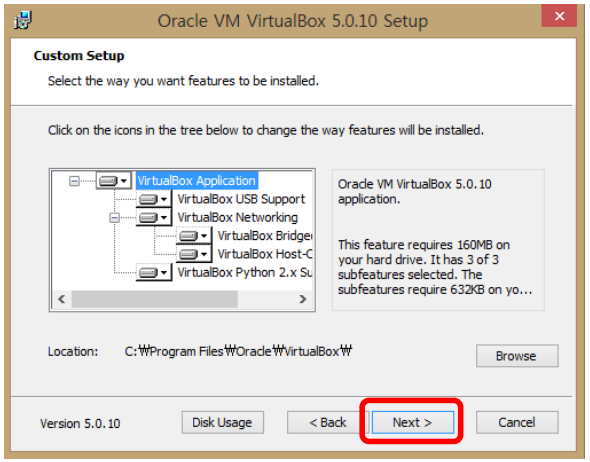
https://www.virtualbox.org

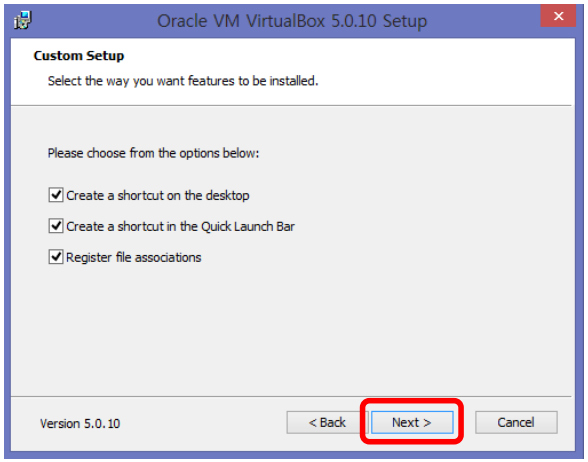
가상 환경 구축(VirtualBox Download)

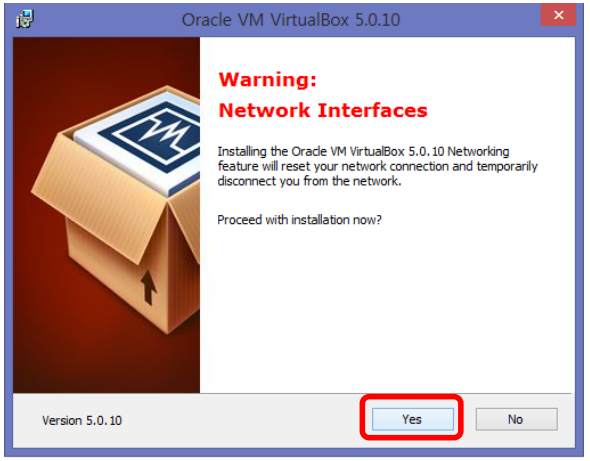
https://www.virtualbox.org

가상 환경 구축(VitualBox 설치)

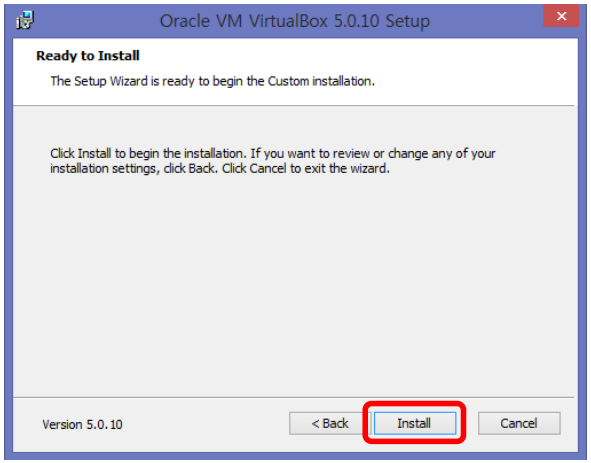


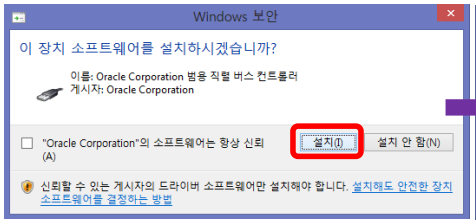


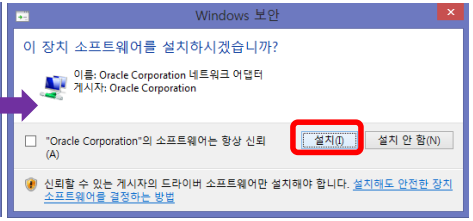


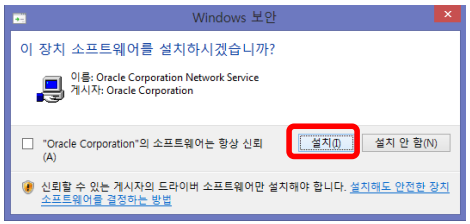


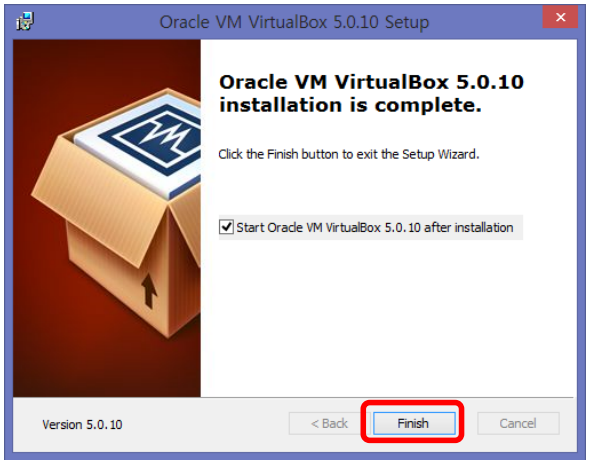
가상 환경 구축(VitualBox 설치)

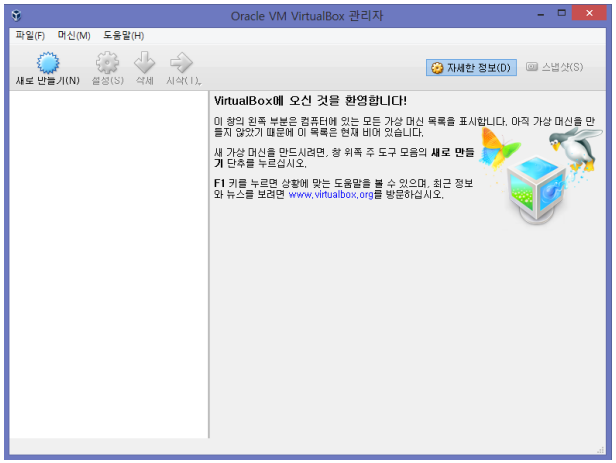






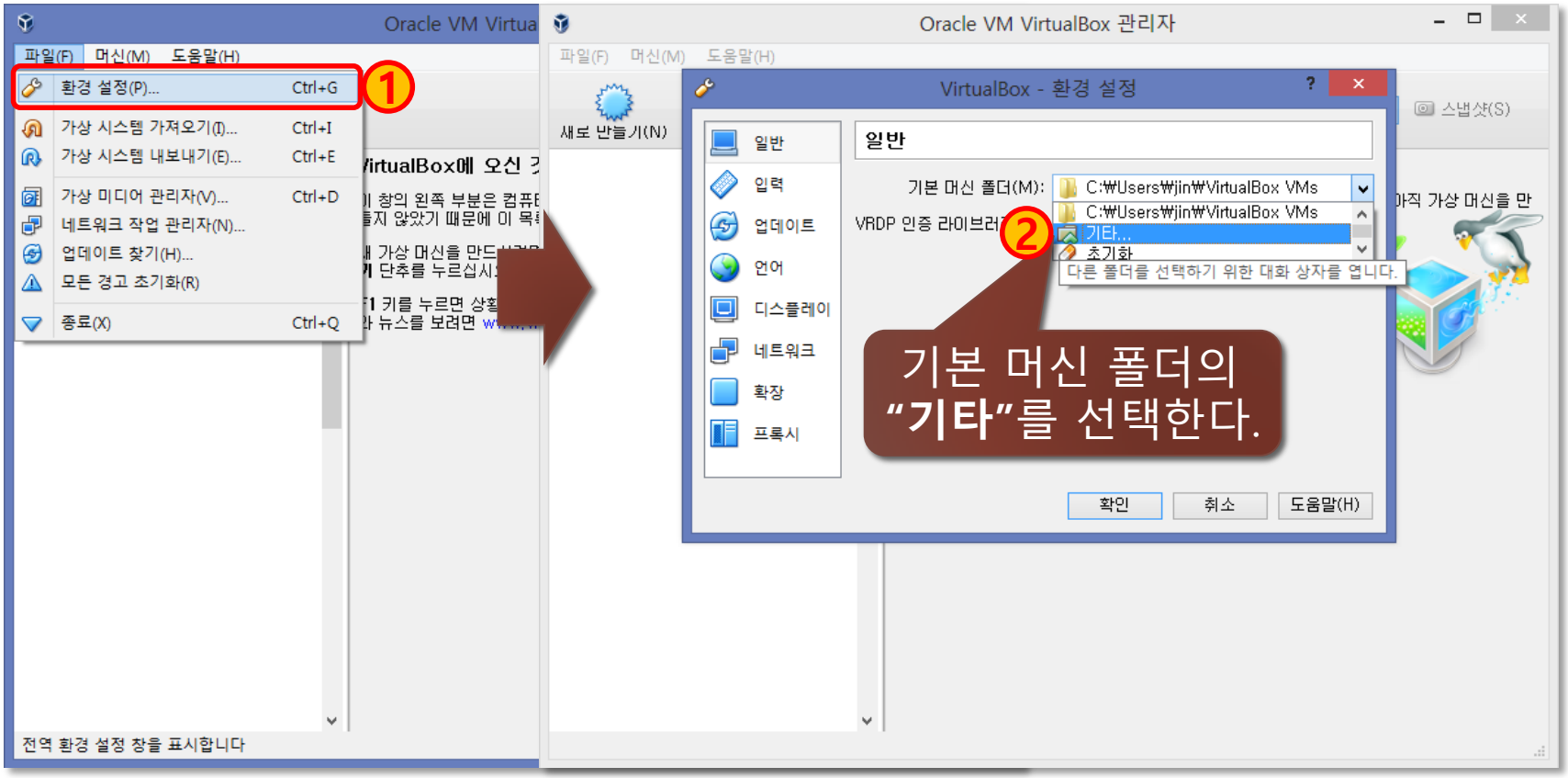






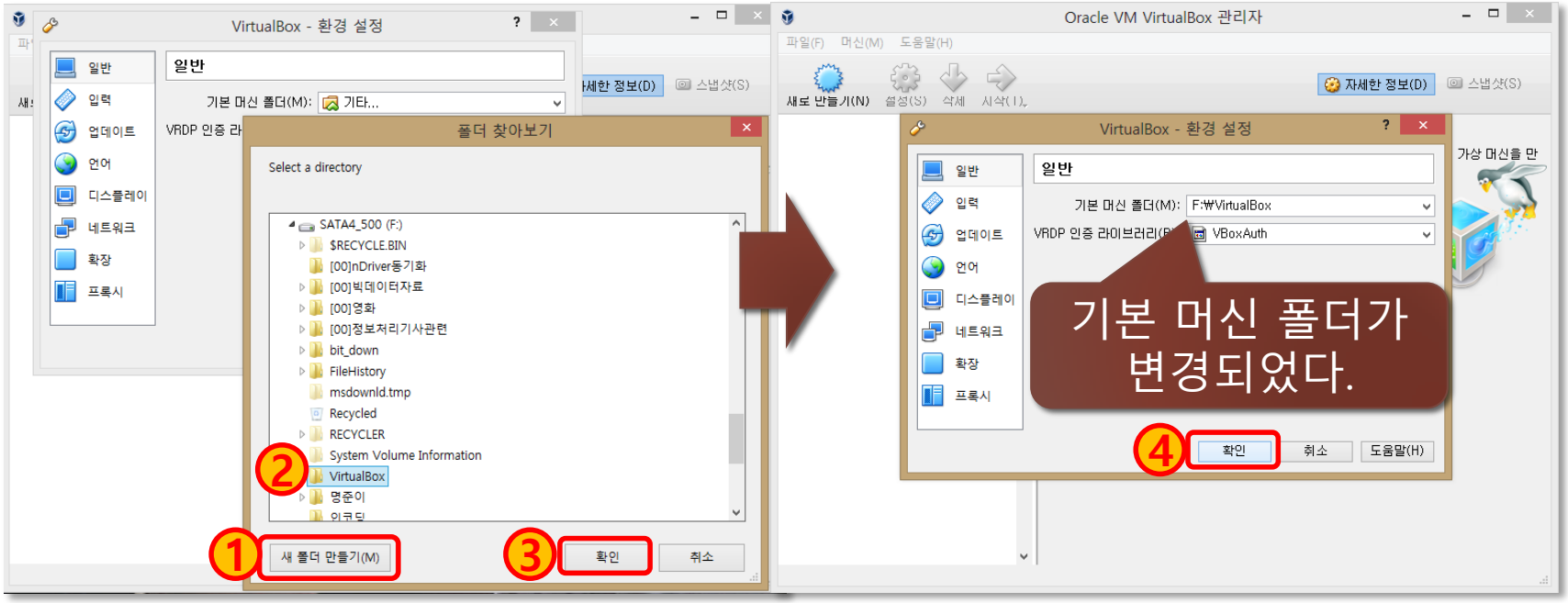
가상 환경 구축(VitualBox 환경설정)

가상 머신 기본 폴더 변경



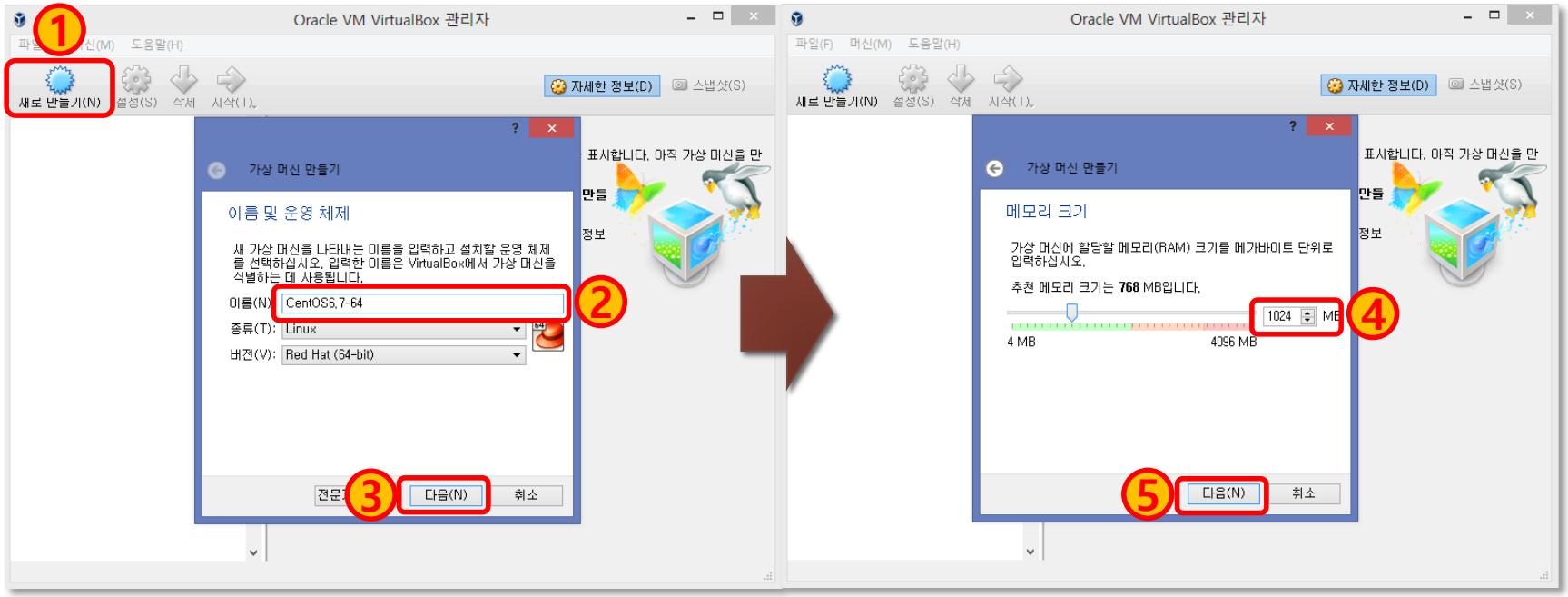
가상 환경 구축(VitualBox 환경설정)

가상 머신 기본 폴더 변경



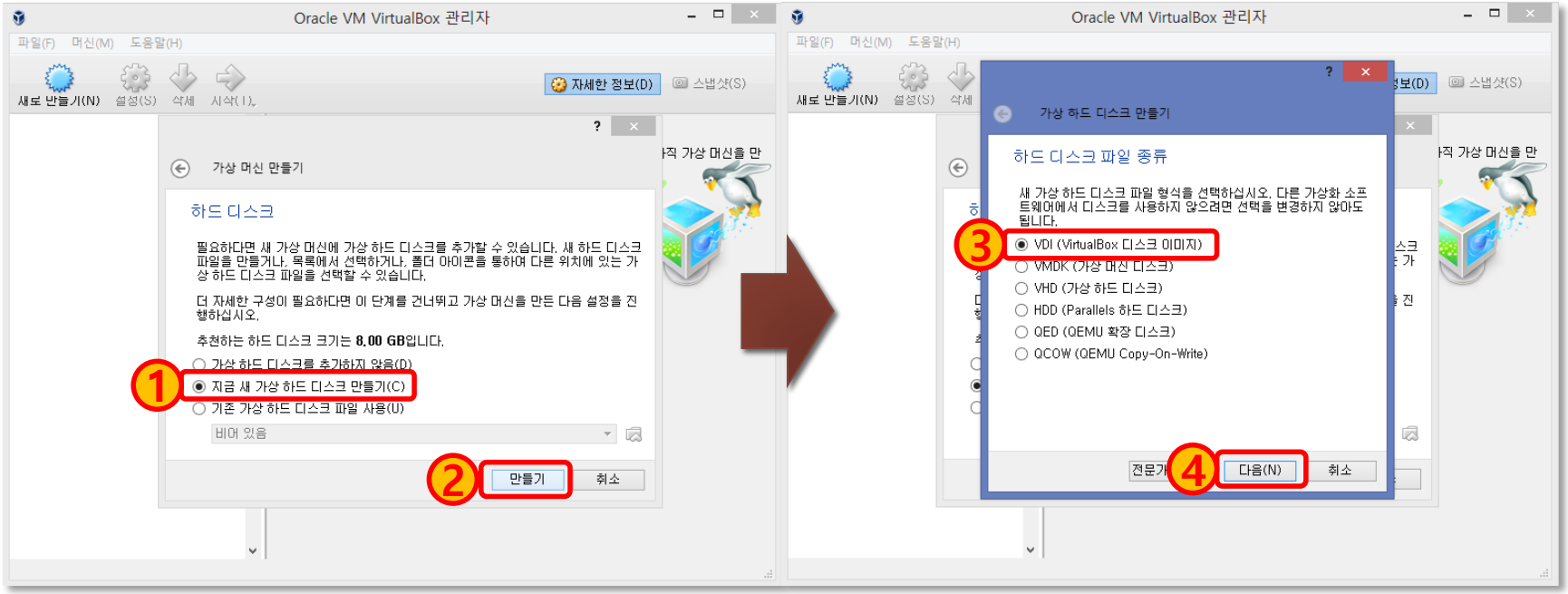
C드라이브가 아닌 D, E, F 중 하나의 드라이브를 선택하고 새 폴더 만들기로 VirtualBox 폴더를 생성한 후 "확인"을 클릭하여 기본 머신 폴더를 설정하고 ④ 번 "확인" 버튼을 클릭하여 설정을 완료 한다.

가상 환경 구축(가상 머신 만들기)



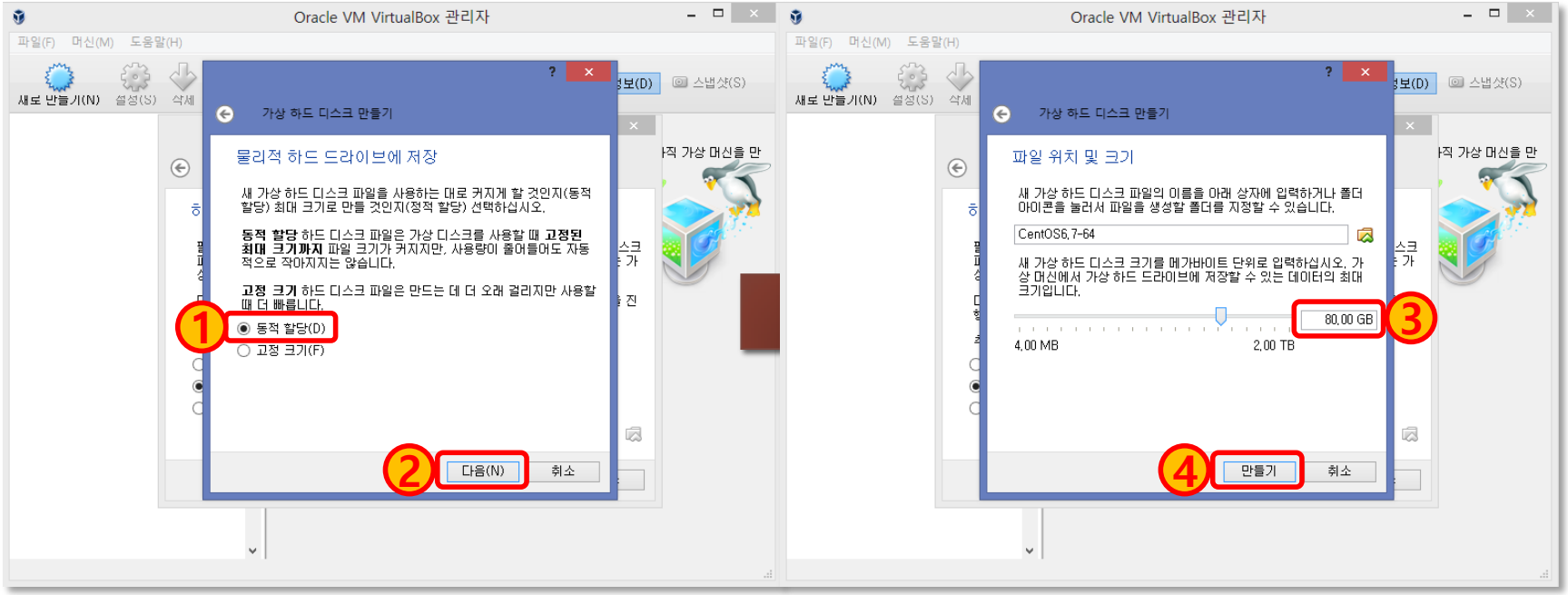
새로만들기 버튼을 클릭하여 나타나는 좌측 그림의 가상 머신 만들기 대화상자에서 "CentOS6.7-64"를 입력하면 종류(T) : Linux, 버전(V) : Red Hat(64-bit)이 자동으로 선택된다. 자동 선택되지 않으면 수동으로 지정한 후 "다음"을 클릭 한다. 우측 그림에서 메모리 크기를 1024MB로 지정하고 "다음"을 클릭한다. 메모리 크기를 가상 머신을 만든 후에도 변경이 가능하다.

가상 환경 구축(가상 머신 만들기)



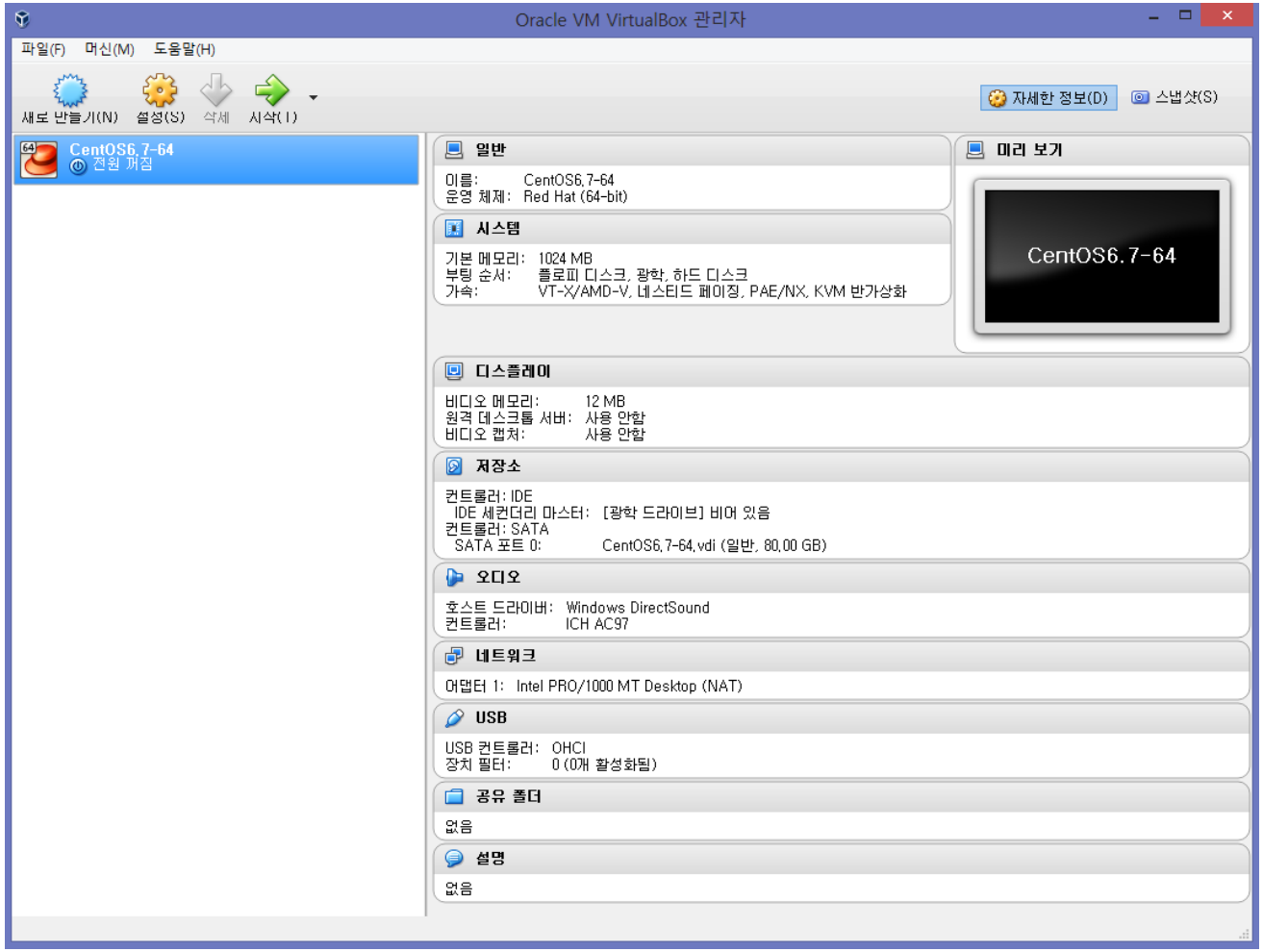
좌측의 하드 디스크 대화상자에서 "지금 새 가상 하드 디스크 만들기(C)"를 선택하고 "만들기"를 클릭한다. 우측의 하드 디스크 파일 종류 선택 대화상자에서 "VDI(VirtualBox 디스크 이미지)"를 선택하고 "다음"을 클릭 한다.

가상 환경 구축(가상 머신 만들기)



좌측의 물리적 하드 드라이브에 저장 대화상자에서 "동적 할당(D)"를 선택하고 "다음"을 클릭 한다. 우측의 파일 위치 및 크기 대화상자에서 하드 디스크 크기를 "80.00GB"로 설정하고 "만들기"를 클릭하여 가상 머신을 생성한다. 하드 디스크 크기를 동적 할당 80GB로 지정했기 때문에 가상 시스템에 설치되는 프로그램의 크기에 따라서 하드 디스크를 차지하는 용량이 변하게 된다. 하드 디스크 크기는 한 번 지정하면 변경할 수 없기 때문에 조금 넉넉히 지정하였다.

가상 환경 구축(가상 머신 만들기 - 완료)

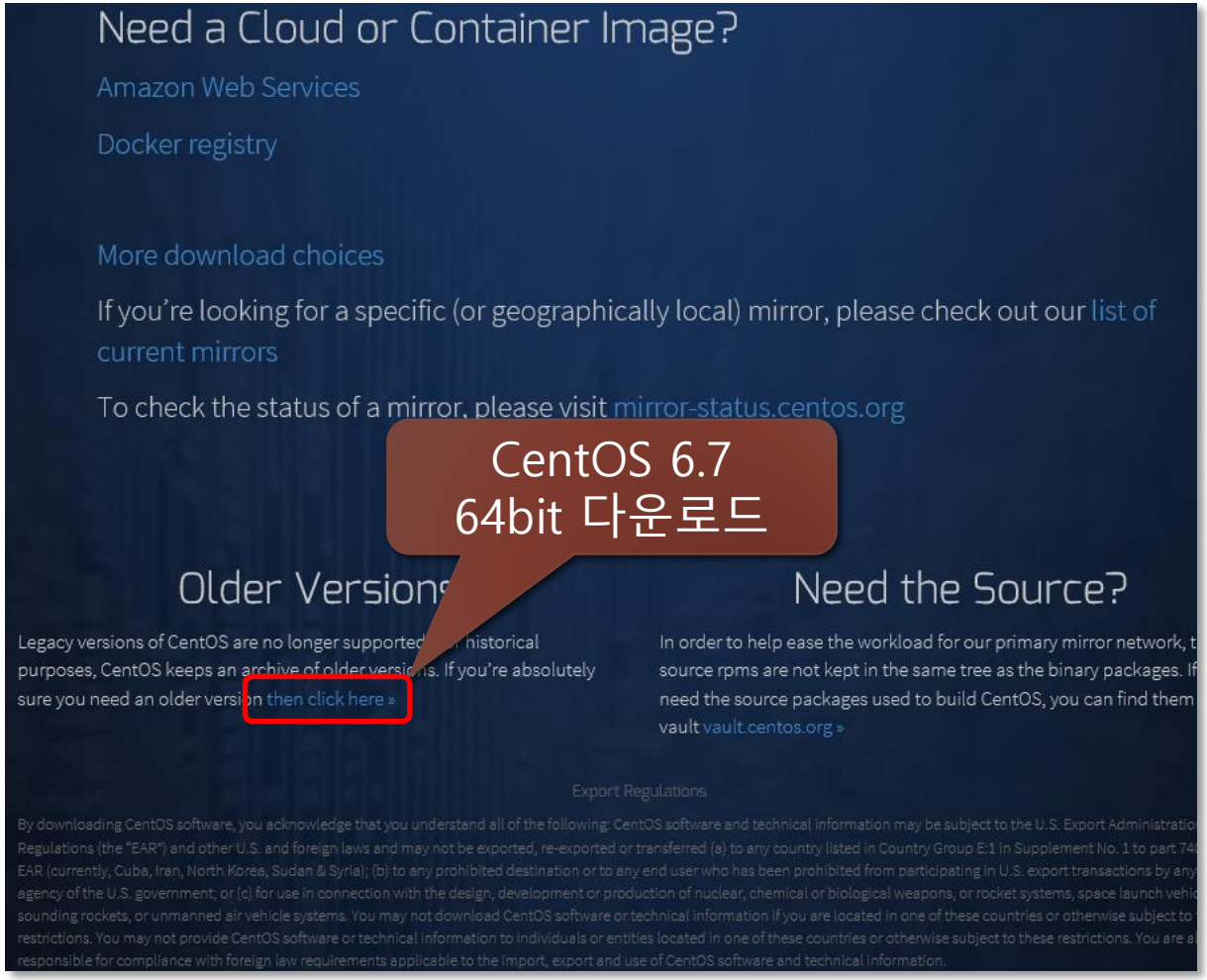


Linux(Cent OS) 다운로드(1)

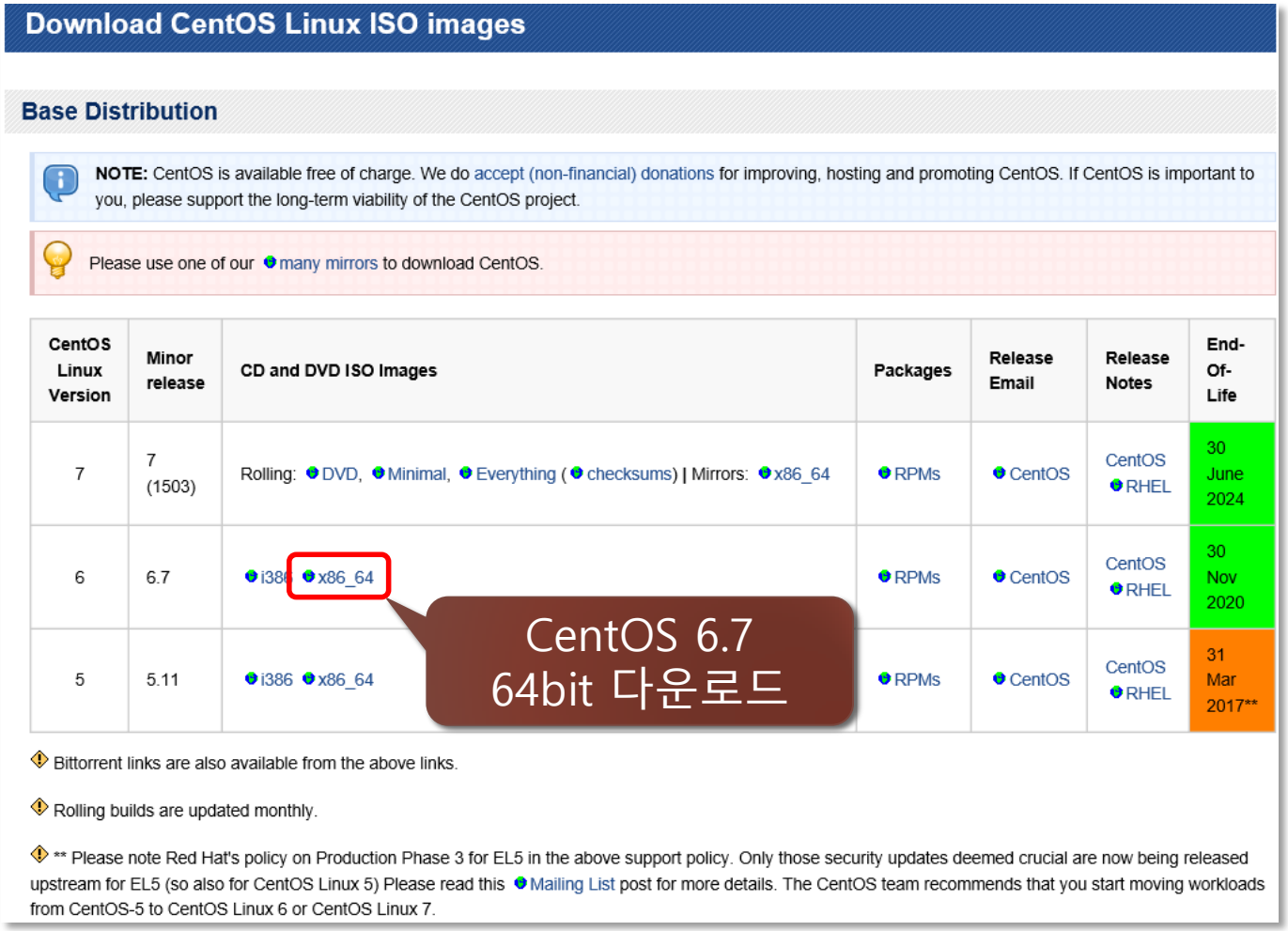
- OS 선택
  - 배포판 : 레드햇(Redhat) 계열의 CentOS
  - 최신 버전 : CentOS-7-x86_64-DVD-1503-01.iso
  - 실습용 버전 : **CentOS-6.7 64bit** 다운로드
  - 다운로드 주소 : **https://www.centos.org/download**
- 실습용 다운로드 파일
  - **CentOS-6.7-x86_64-bin-DVD1.iso**
- 고려사항
  - 리눅스는 어떤 배포판과 버전을 사용할 것인가?
  - 32bit는 i386, 64bit는 x86_64을 다운로드
  - 하둡 등의 설치패키지인 경우에도 32bit와 64bit를 구분해 설치

Linux(Cent OS) 다운로드(2)

Linux(Cent OS) 다운로드(3)



Linux(Cent OS) 다운로드(4)



Download CentOS Linux ISO images

Base Distribution

NOTE: CentOS is available free of charge. We do accept (non-financial) donations for improving, hosting and promoting CentOS. If CentOS is important to you, please support the long-term viability of the CentOS project.

Please use one of our many mirrors to download CentOS.

CentOS Linux Version	Minor release	CD and DVD ISO Images	Packages	Release Email	Release Notes	End-Of-Life
7	7 (1503)	Rolling: DVD, Minimal, Everything (checksums) | Mirrors: x86_64	RPMs	CentOS	CentOS RHEL	30 June 2024
6	6.7	i386 x86_64	RPMs	CentOS	CentOS RHEL	30 Nov 2020
5	5.11	i386 x86_64	RPMs	CentOS	CentOS RHEL	31 Mar 2017**

CentOS 6.7
64bit 다운로드

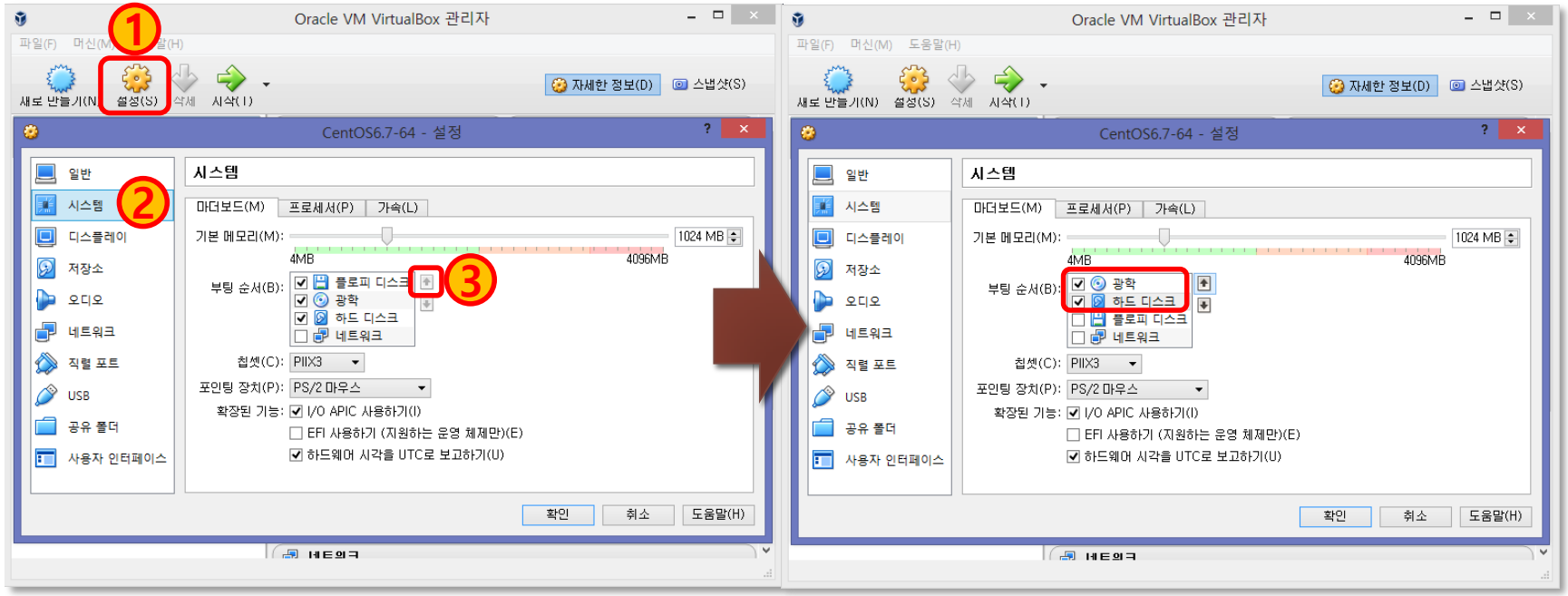
Bittorrent links are also available from the above links.

Rolling builds are updated monthly.

** Please note Red Hat's policy on Production Phase 3 for EL5 in the above support policy. Only those security updates deemed crucial are now being released upstream for EL5 (so also for CentOS Linux 5) Please read this Mailing List post for more details. The CentOS team recommends that you start moving workloads from CentOS-5 to CentOS Linux 6 or CentOS Linux 7.

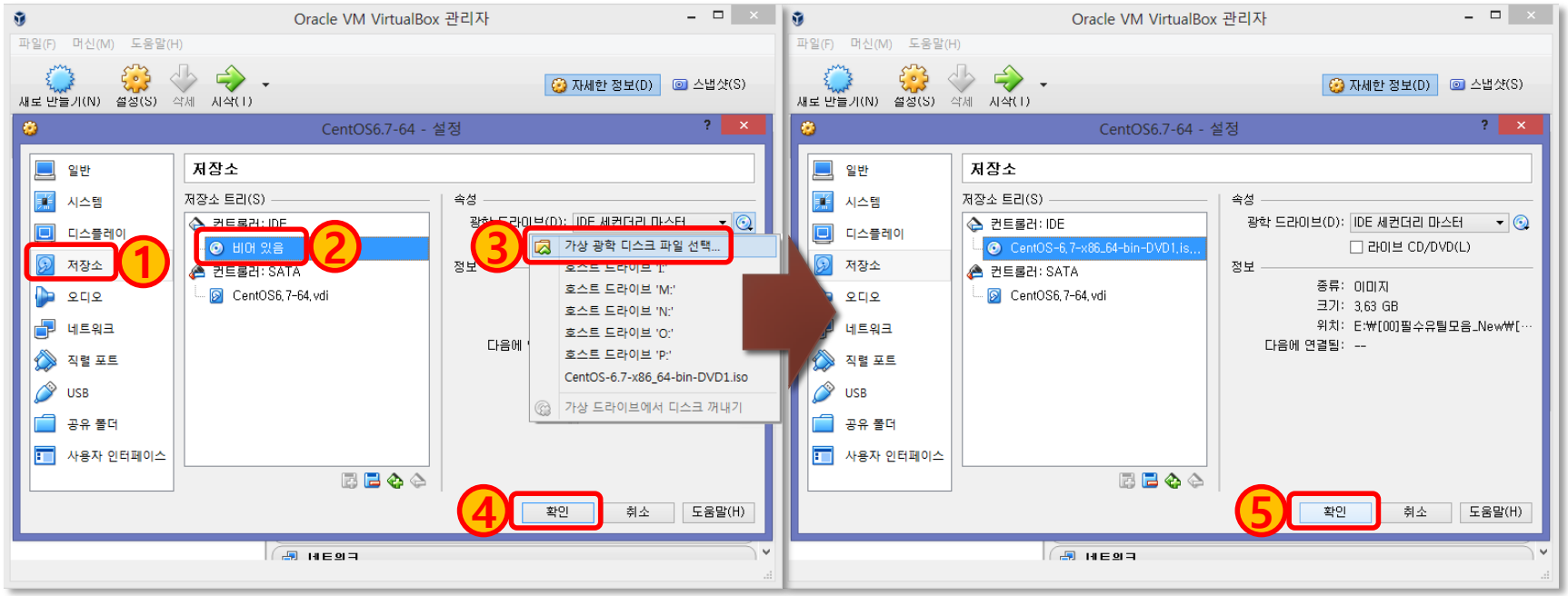
Linux(Cent OS) 다운로드(5)

Linux(Cent OS) 설치 – 부팅 순서 설정



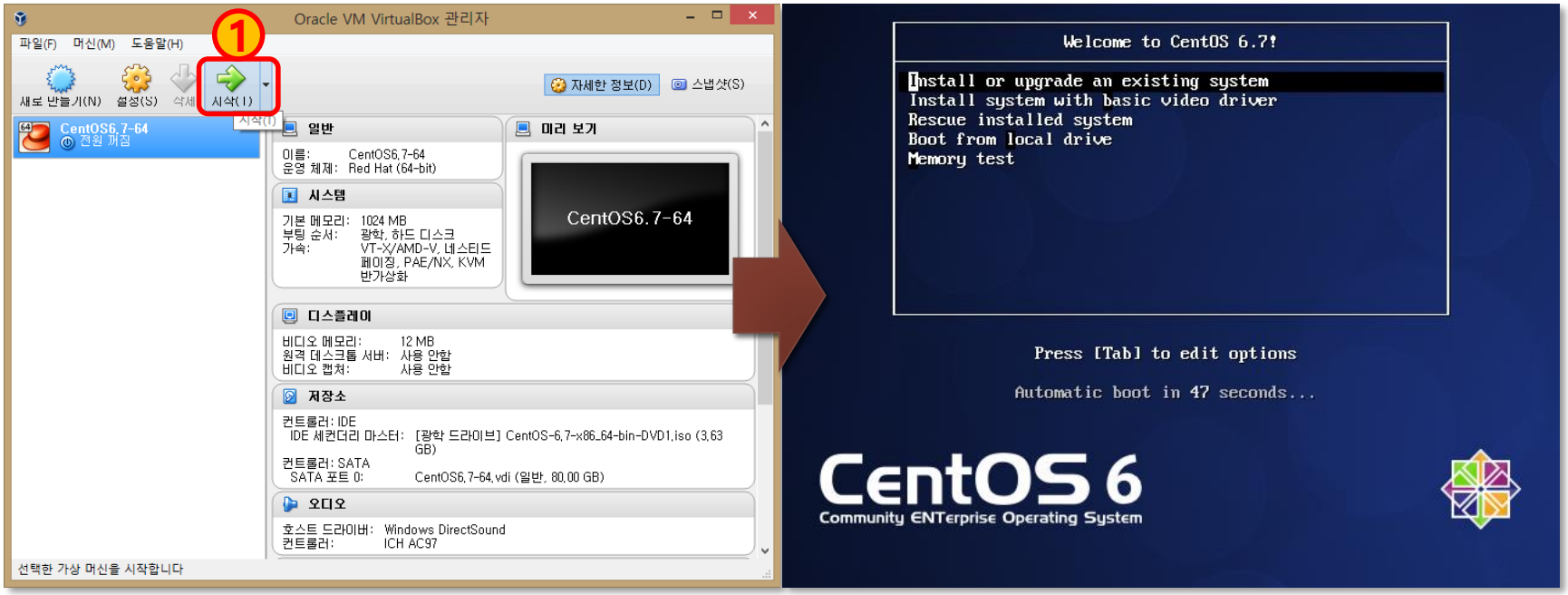
VirtualBox를 실행하여 앞에서 만들었던 "CentOS6.7-64" 가상 머신을 선택하고 좌측 그림 ①번과 같이 "설정" 단축 아이콘을 클릭하여 나타나는 가상 머신 설정 대화상자에서 ②번 "시스템"을 선택하고 "부팅 순서(B)"에서 "플로피 디스크"의 선택을 해제하고 "광학"을 ③번 화살표 버튼을 이용해 우측 그림과 같이 맨 위로 이동 시키고 "하드 디스크"를 두 번째로 이동 시킨다.

Linux(Cent OS) 설치 – 설치 디스크 지정



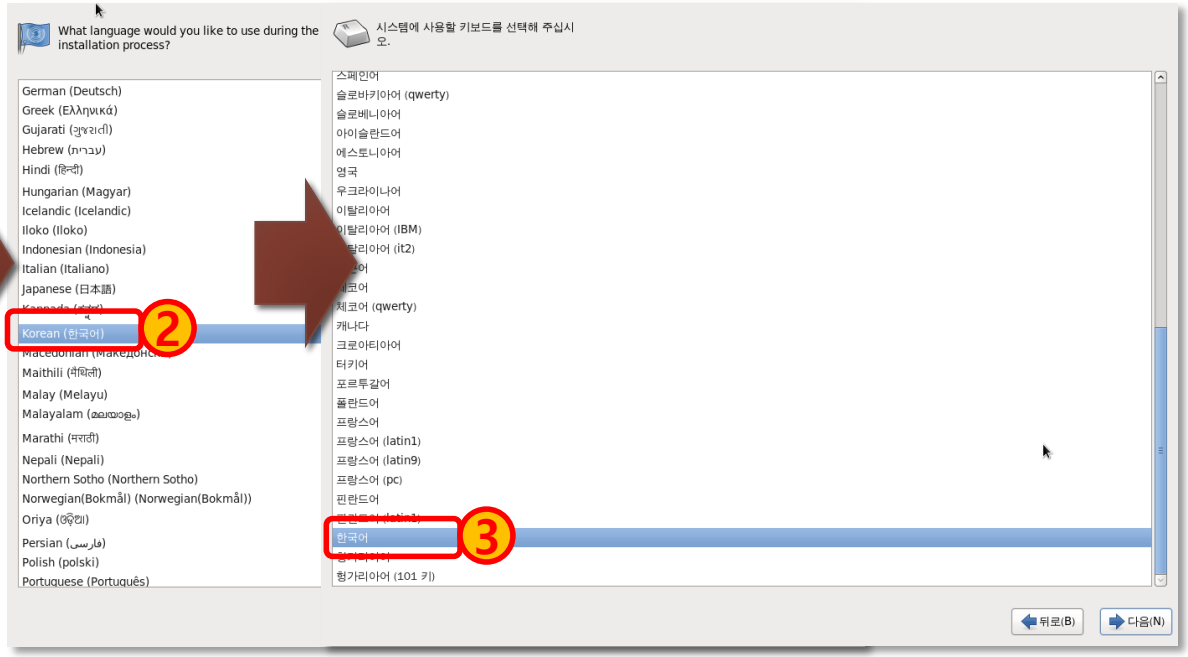
좌측 그림에서 "저장소"를 선택하고 "저장소 트리(S)"에서 "컨트롤러: IDE"를 선택 광학 드라이브(D)의 디스크 모양 아이콘을 클릭하여 다운로드 받은 "CentOS-6.7-x86_64-bin-DVD1.iso" 파일을 선택하고 "확인"을 클릭 한다. "저장소 트리(S)"에서 "컨트롤러: IDE"에 "CentOS-6.7-x86_64-bin-DVD1.iso" 파일이 제대로 선택된 것을 확인하고 "확인"을 클릭한다.

Linux(Cent OS) 설치 – 가상 머신 시작



좌측 그림에서 "CentOS6.7-64" 가상 머신을 선택하고 ① 번 "시작" 아이콘을 클릭하면 가상 머신이 시작되고 광학 드라이브에 지정한 CentOS6.7이 실행되면서 우측의 그림과 같이 설치 옵션을 선택하는 화면이 나타난다. "CentOS6.7-64" 가상 머신을 더블클릭 해도 동일한 동작을 한다.

Linux(Cent OS) 설치 – 언어, 키보트 설정

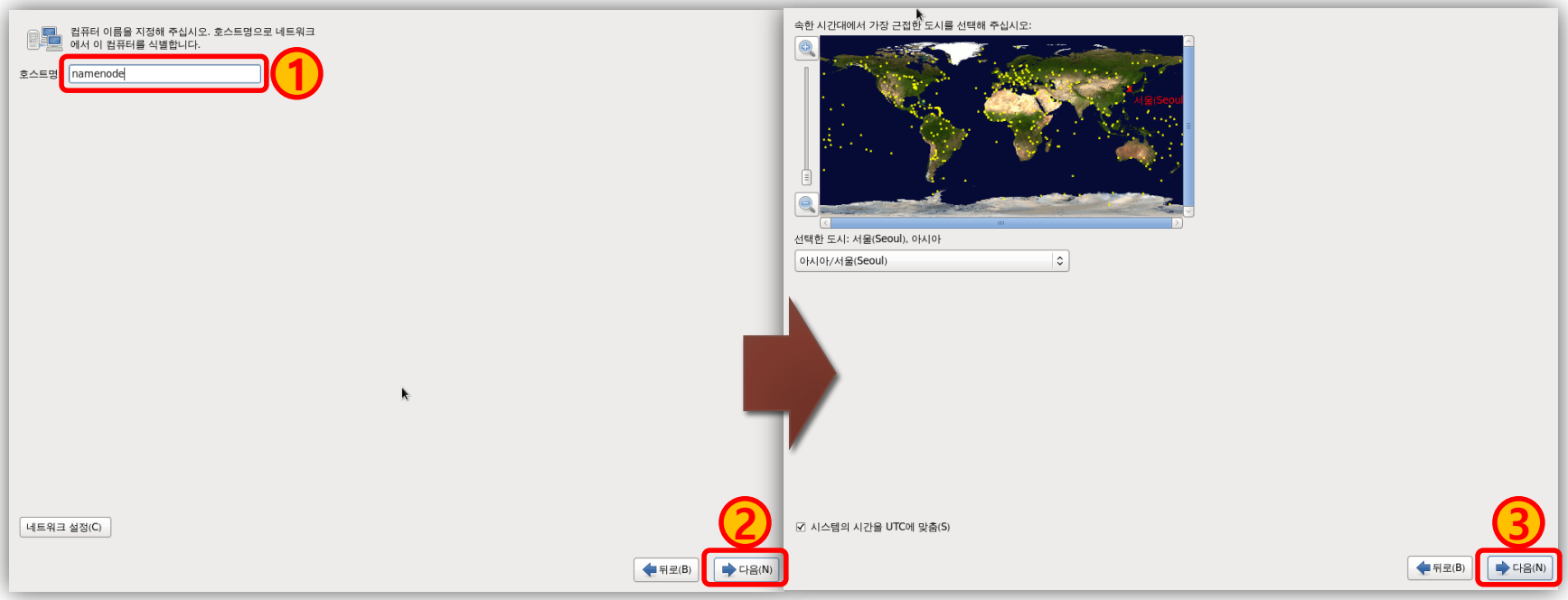


좌측 그림에서 방향 키를 이용해 "Skip"을 선택하고 "Enter" 키를 누르면 CentOS6 로고 화면이 나타나는데 이 화면에서 "Next" 버튼을 클릭한다. 그럼 ②번과 같이 언어 선택 창이 나타나는데 이 화면에서 "Korean(한국어)"를 선택하고 "Next" 버튼을 클릭하면 ③번과 같이 시스템에 사용할 키보드를 선택하는 창이 나타난다. 이 창에서 "한국어"를 선택하고 "다음(N)" 버튼을 클릭 한다.

Linux(Cent OS) 설치 – 기본 저장 장치

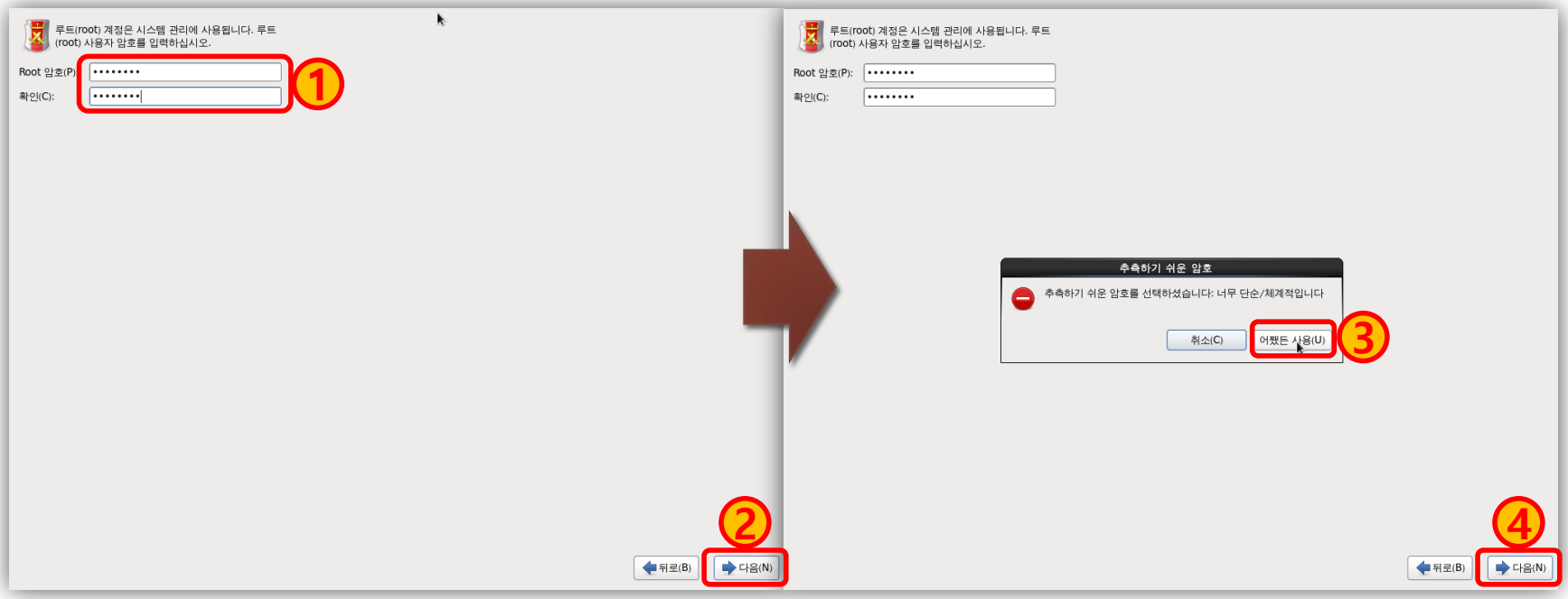
좌측 그림의 저장장치 경고 대화상자에서 "예, 모든 데이터를 삭제합니다(Y)"를 클릭한 후 "다음(N)"을 클릭하여 우측의 그림과 같은 대화상자가 나타나면 "기본 저장 장치"를 선택하고 "다음(N)" 버튼을 클릭해 다음 단계로 넘어간다.

Linux(Cent OS) 설치 – 호스트명, 지역 시간대 설정



좌측 그림과 같이 호스트명 입력란에 "namenode"를 입력하고 "다음" 버튼을 클릭하면 우측 그림과 같이 지역 시간대를 선택할 수 있는 화면이 나타난다. 이 화면에서 "아시아/서울(Seoul)"을 선택하고 "다음" 버튼을 클릭해 다음 단계로 넘어간다.

Linux(Cent OS) 설치 – root 계정 암호 설정



좌측 그림과 같이 시스템 관리자인 root 계정의 암호를 "12345678"로 입력하고 "다음" 버튼을 클릭하면 우측 그림과 같이 "추측하기 쉬운 암호" 창이 나타나는데 ③번과 같이 "어쨌든 사용(U)" 버튼을 클릭하고 ④번 "다음" 버튼을 클릭해 다음 단계로 넘어간다.

Linux(Cent OS) 설치 – 하드 디스크 파티션

파티션이란 하나의 물리적인 하드디스크를 논리적으로 여러 개 나누어 사용하는 것을 말하며 파티션의 파일 시스템 방식은 운영체제에 따라 다르다.

윈도우즈

- 파티션을 나누어 C: 또는 D: 와 같이 드라이브로 사용
- 파일 시스템 방식
  윈도우95/98/ME : FAT32
  윈도우NT/2000/7/8 : NTFS

리눅스

- 파티션을 나누어 디렉터리와 연결해 사용
- 하드디스크를 논리적인 파티션으로 나눈 후 포맷하여 파일시스템 구축
- 파일 시스템 방식
  ext2, ext3, ext4 등

Linux(Cent OS) 설치 – 하드 디스크 파티션

파티션 종류

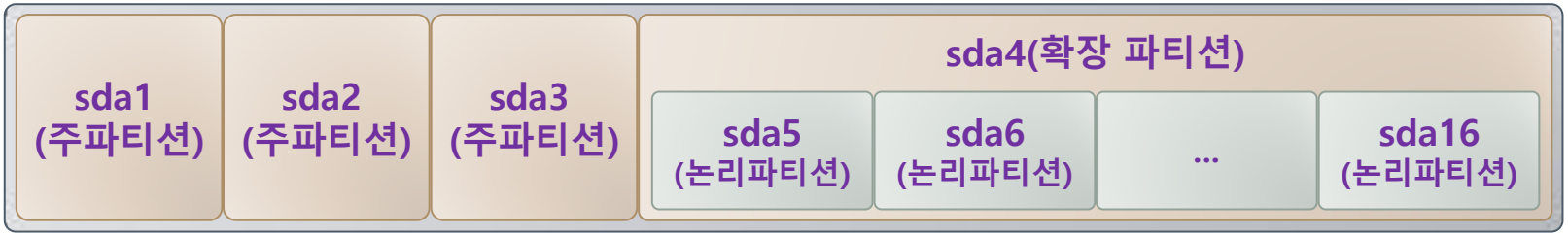
- **주 파티션(Primary Partition)**
  기본 파티션으로 물리적 디스크 하나 당 주 파티션은 최대 3개까지 나누어 사용할 수 있다.

- **확장 파티션(Extend Partition)**
  확장 파티션은 물리적 디스크 하나에 확장 파티션은 한 개만 사용할 수 있으며 저장공간을 갖지 않고 논리 파티션을 담는 역할을 한다.

- **논리 파티션(Logical Partition)**
  하나의 물리적이 하드디스크에 4개 이상의 파티션을 사용하기 위해서는 확장 파티션을 만들고 그 안에 논리 파티션을 나누어 사용하면 된다. 논리 파티션은 하드디스크 하나에 최대 12개 까지 나누어 사용할 수 있다.

Linux(Cent OS) 설치 – 하드 디스크 파티션

▪ 하드디스크 파티션 구성

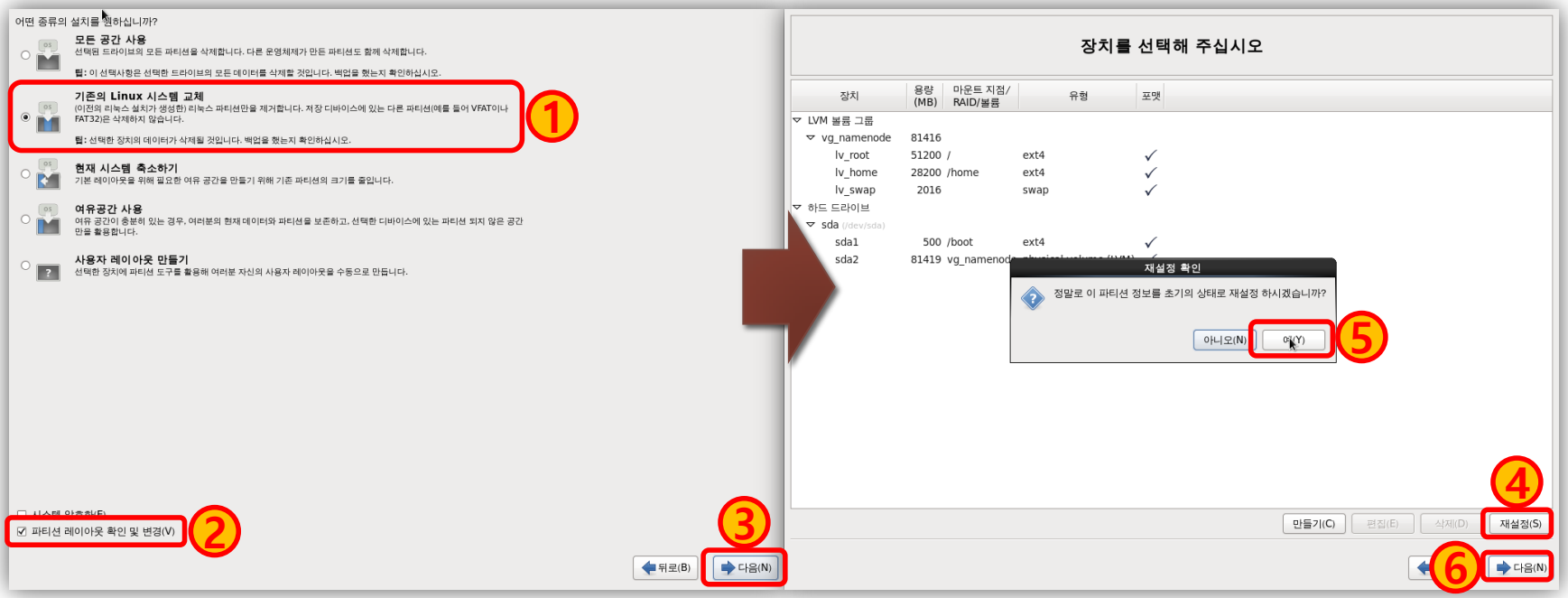


▪ 가상 머신 하드디스크 파티션 구성(80GB)

마운트 지정	디렉터리	용 량	비 고
/boot	/dev/sda1	1024MB(1GB)	부팅 커널 저장
/	/dev/sda2	20480MB(20GB)	root 파티션
/var	/dev/sda3	10240MB(10GB)	로그, 캐시 파일 등을 저장
swap	/dev/sda5	2048MB(2GB)	RAM 부족 시 사용, RAM의 2배 지정
/home	/dev/sda6	48125(47GB)	사용자 별 공간
/tmp			임시파일 저장
/usr			응용프로그램이 저장

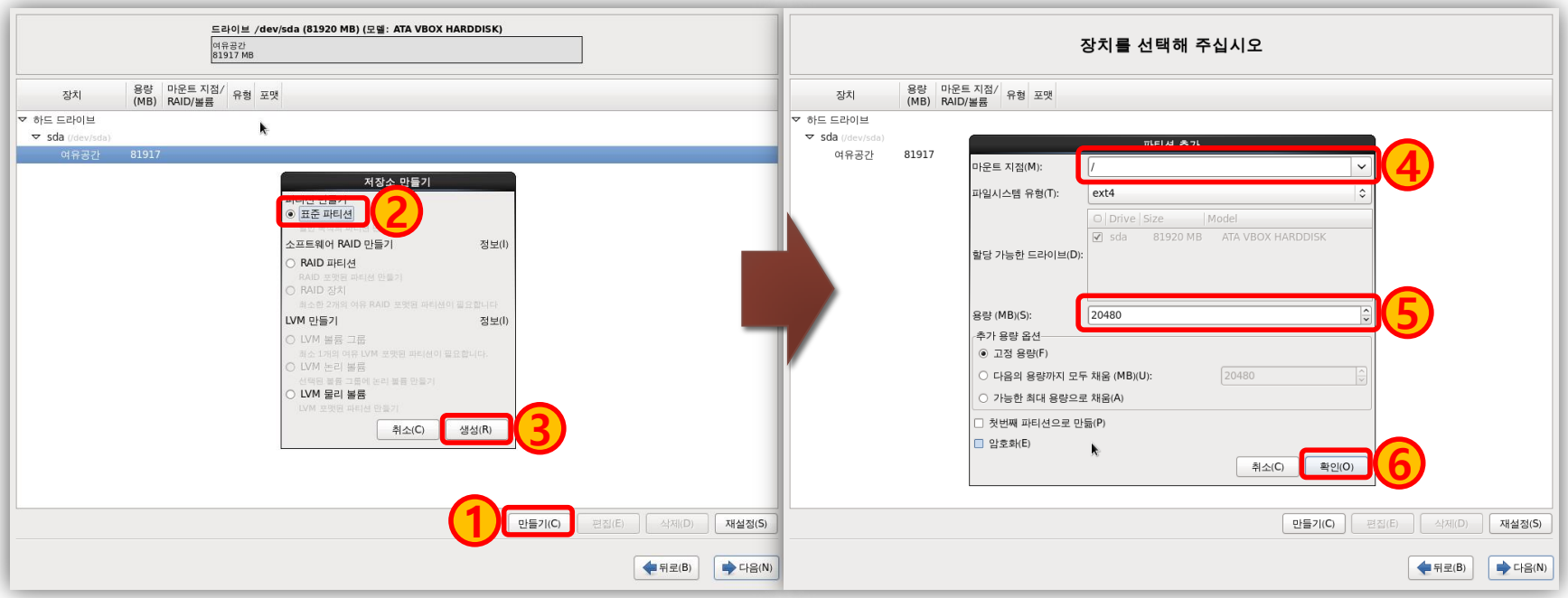
➢ root(/) 파티션과 swap 파티션만 생성하면 나머지는 / 파티션에 생성된다.

Linux(Cent OS) 설치 – 파티션 삭제



좌측 그림에서와 같이 ①번 "기존 Linux 시스템 교체"를 선택하고 ②번 "파티션 레이아웃 확인 및 변경(V)"을 체크한 후 "다음" 버튼을 클릭하면 우측 그림과 같은 화면이 나타난다. 이 화면에서 ④번 "재설정(S)" 버튼을 클릭해 나타나는 ⑤번 재설정 확인 창에서 "예(Y)" 버튼을 클릭하면 기존의 모든 파티션이 지워진다. 그리고 ⑥번 "다음" 버튼을 클릭해 다음 단계로 넘어간다.

Linux(Cent OS) 설치 – root(/) 파티션 생성



좌측 그림에서 ①번 "만들기" 버튼을 클릭하여 "저장소 만들기" 대화상자가 나타나면 ②번 "표준 파티션"을 선택하고 "생성" 버튼을 클릭하면 우측 그림과 같이 "파티션 추가" 대화상자가 나타난다. 이 화면에서 ④번 ▼ 버튼을 클릭해 파티션 리스트에서 root(/) 파티션을 선택한 후 이 "/" 파티션의 용량을 ⑤번과 같이 20480MB(20GB)로 지정하고 ⑥번 "확인" 버튼을 클릭해 root(/) 파티션을 생성 한다.

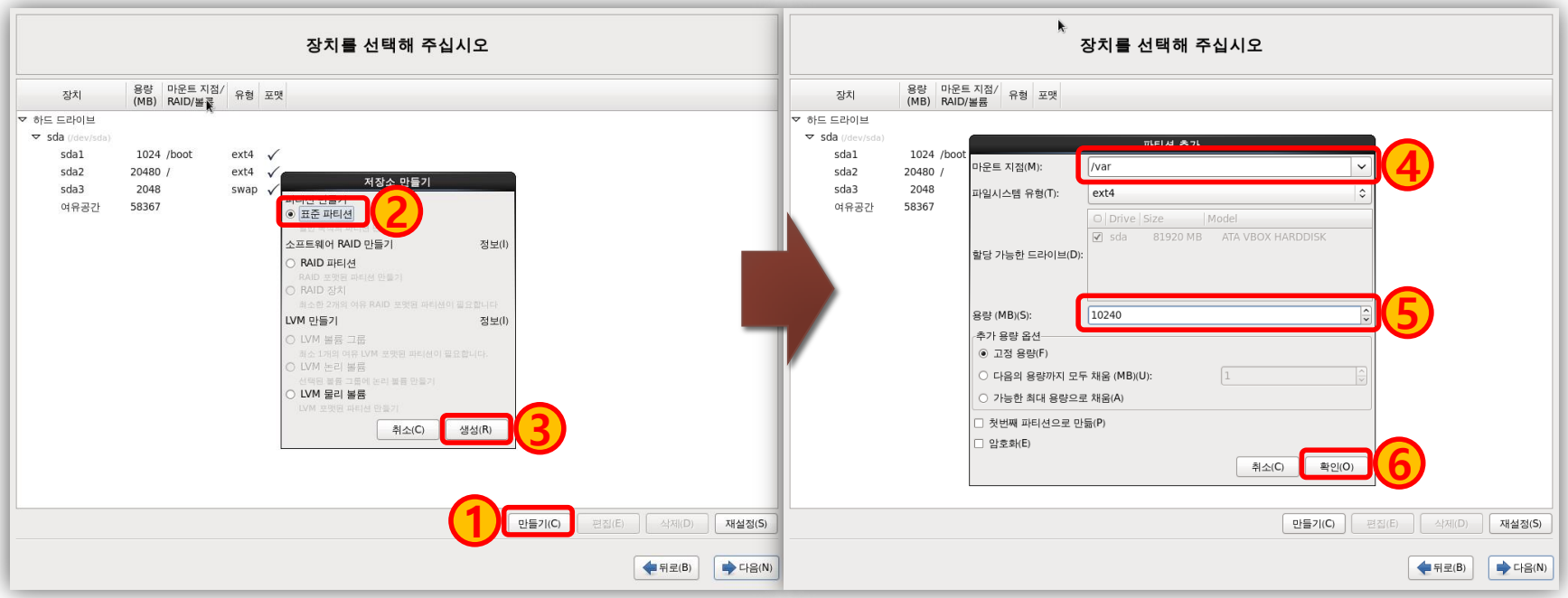
Linux(Cent OS) 설치 – /boot 파티션 생성

좌측 그림에서 ①번 "만들기" 버튼을 클릭하여 "저장소 만들기" 대화상자가 나타나면 ②번 "표준 파티션"을 선택하고 "생성" 버튼을 클릭하면 우측 그림과 같이 "파티션 추가" 대화상자가 나타난다. 이 화면에서 ④번 ▼ 버튼을 클릭해 파티션 리스트에서 /boot 파티션을 선택한 후 이 /boot 파티션의 용량을 ⑤번과 같이 1024MB(1GB)로 지정하고 ⑥번 "확인" 버튼을 클릭해 /boot 파티션을 생성 한다.

Linux(Cent OS) 설치 – swap 파티션 생성

좌측 그림에서 ①번 "만들기" 버튼을 클릭하여 "저장소 만들기" 대화상자가 나타나면 ②번 "표준 파티션"을 선택하고 "생성" 버튼을 클릭하면 우측 그림과 같이 "파티션 추가" 대화상자가 나타난다. 이 화면에서 ④번 ▼ 버튼을 클릭해 파일시스템 유형을 "swap"으로 선택한 후 용량을 ⑤번과 같이 2048MB(2GB, 일반적으로 RAM의 2배로 지정)로 지정하고 ⑥번 "확인" 버튼을 클릭해 swap 파티션을 생성 한다.

Linux(Cent OS) 설치 – /var 파티션 생성

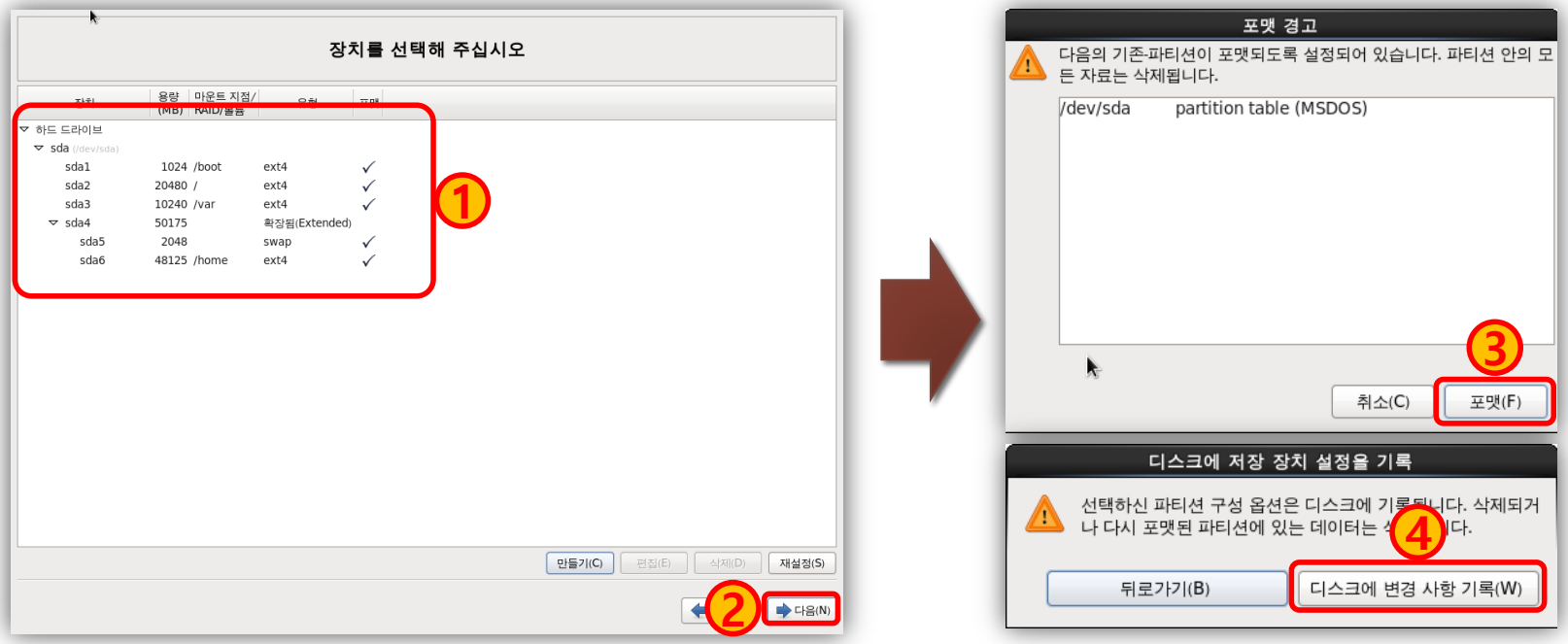


좌측 그림에서 ①번 "만들기" 버튼을 클릭하여 "저장소 만들기" 대화상자가 나타나면 ②번 "표준 파티션"을 선택하고 "생성" 버튼을 클릭하면 우측 그림과 같이 "파티션 추가" 대화상자가 나타난다. 이 화면에서 ④번 ▼ 버튼을 클릭해 파티션 리스트에서 /var 파티션을 선택한 후 이 /var 파티션의 용량을 ⑤번과 같이 10240MB(10GB)로 지정하고 ⑥번 "확인" 버튼을 클릭해 /var 파티션을 생성 한다.

Linux(Cent OS) 설치 – /home 파티션 생성

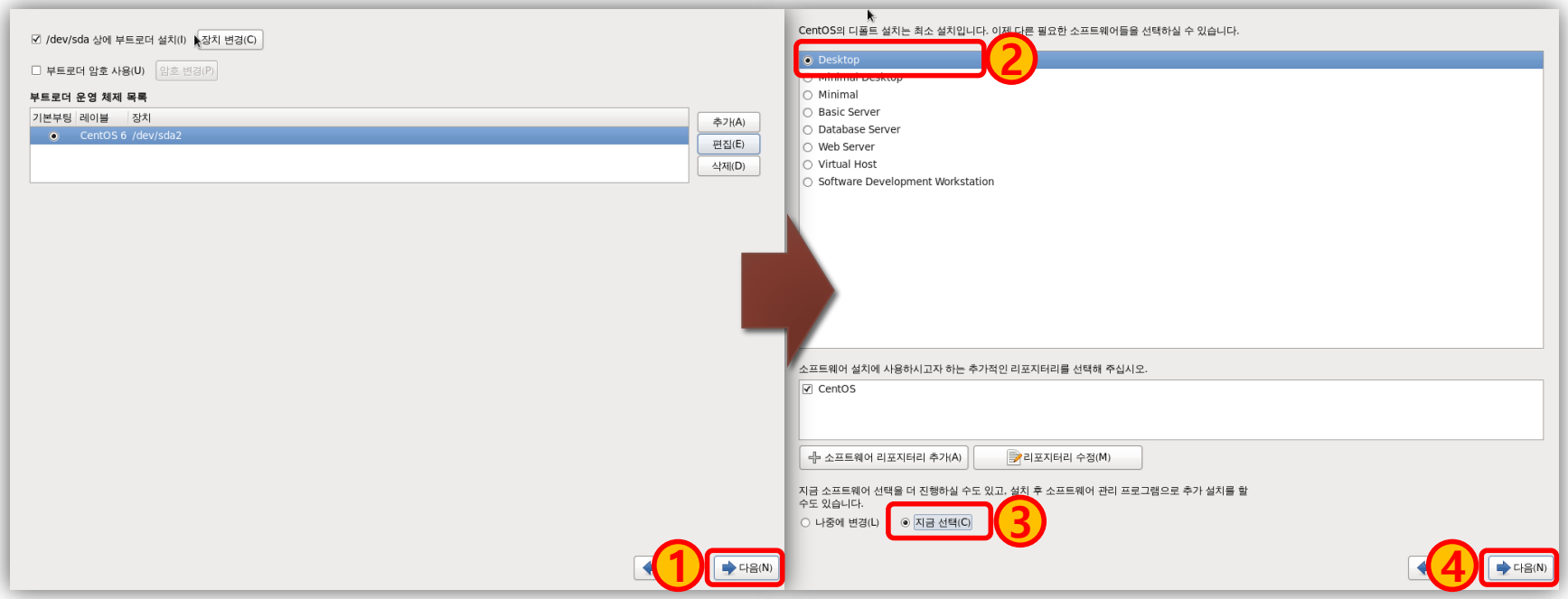
좌측 그림에서 ①번 "만들기" 버튼을 클릭하여 "저장소 만들기" 대화상자가 나타나면 ②번 "표준 파티션"을 선택하고 "생성" 버튼을 클릭하면 우측 그림과 같이 "파티션 추가" 대화상자가 나타난다. 이 화면에서 ④번 ▼ 버튼을 클릭해 파티션 리스트에서 /home 파티션을 선택한 후 이 /home 파티션의 용량은 따로 지정하지 않고 ⑤번과 같이 "가능한 최대 용량으로 채움(A)"을 선택하고 ⑥번 "확인(O)" 버튼을 클릭해 /home 파티션을 생성 한다.

Linux(Cent OS) 설치 – 파티션 설정 완료



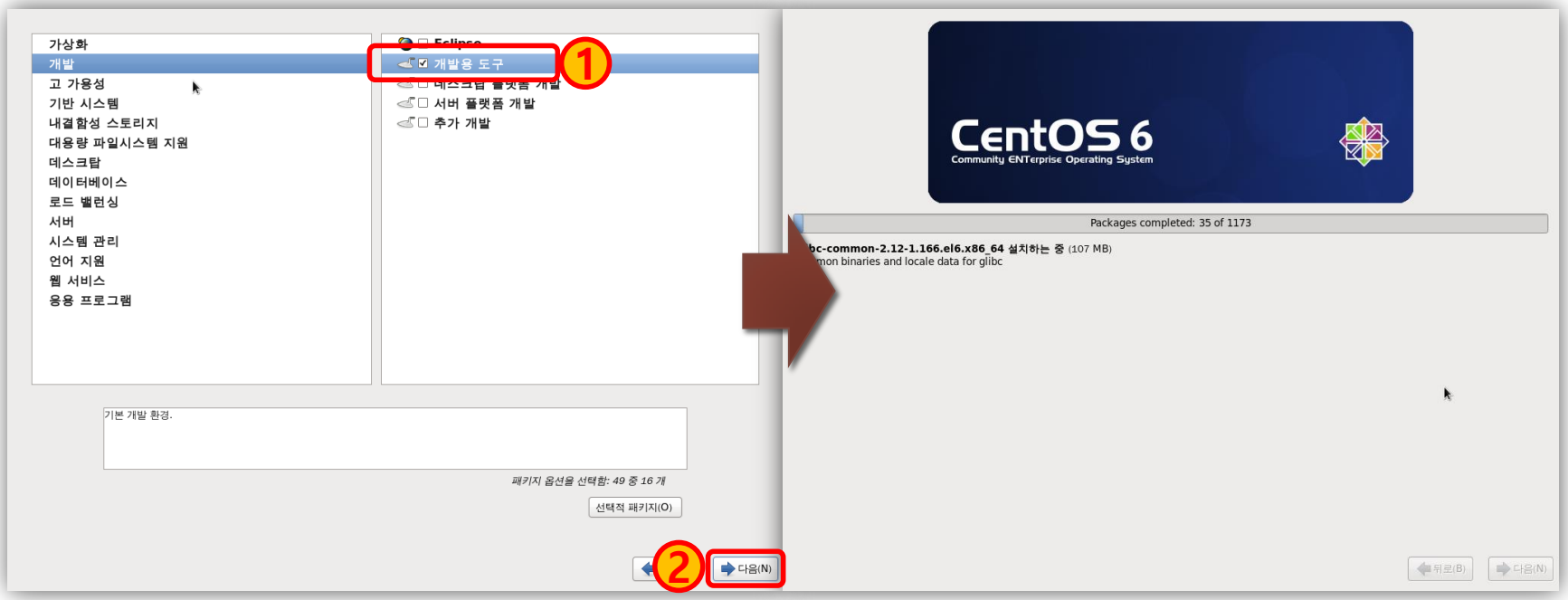
좌측 그림에서 ①번 파티션 설정 내용을 확인하고 "다음" 버튼을 클릭하면 우측 그림에서 "포맷 경고"와 같은 대화상자가 나타날 때가 있는데 "포맷(F)" 버튼을 클릭해 파티션을 포맷하면 된다. 그리고 "디스크에 저장 장치 설정을 기록" 대화상자가 나타나면 "디스크에 변경 사항 기록(W)" 버튼을 클릭해 다음 단계로 넘어간다.

Linux(Cent OS) 설치 – 설치 소프트웨어 선택(1)



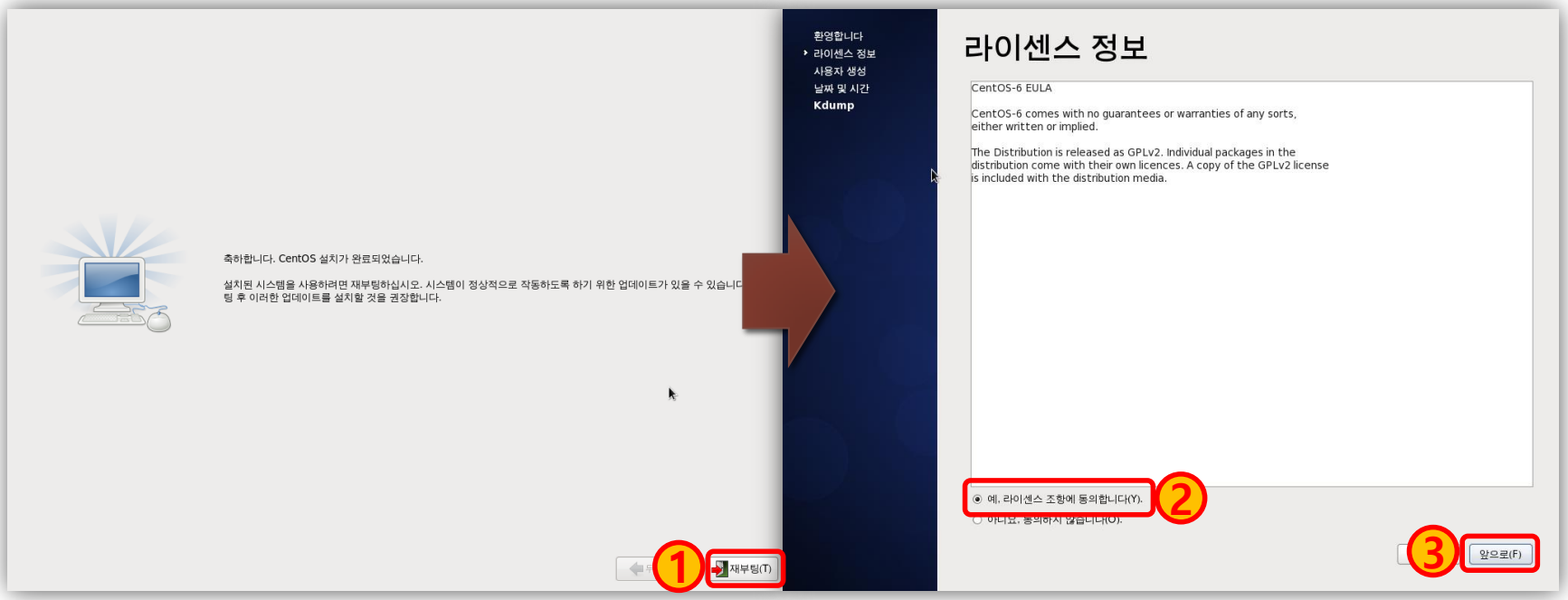
좌측 그림에서 "다음" 버튼을 클릭하면 우측 그리과 같이 설치할 소프트웨어를 선택할 수 있는 대화상자가 나타난다. 이 화면에서 ② 번 "Desktop"을 선택하고 ③ 번 "지금 선택" 선택한 후 "다음(N)" 버튼을 클릭 한다.

Linux(Cent OS) 설치 – 설치 소프트웨어 선택(2)



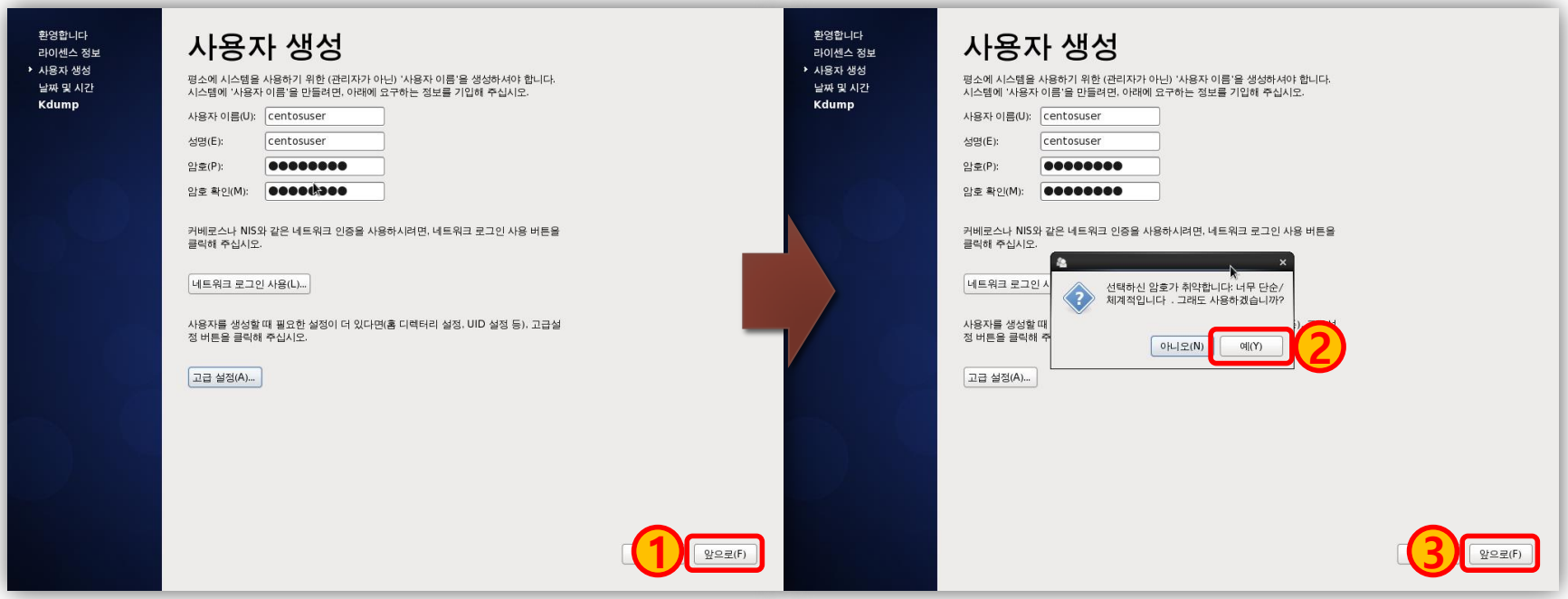
좌측 그림에서 ① 번과 같이 "개발용 도구"를 선택 한다. 더 필요한 소프트웨어가 있다면 각 항목을 클릭해 목록을 확인하여 선택하고 "다음(N)" 버튼을 클릭하면 우측 그림과 같이 CentOS가 가상 시스템에 설치된다.

Linux(Cent OS) 설치 – 설치 완료 및 저작권 동의



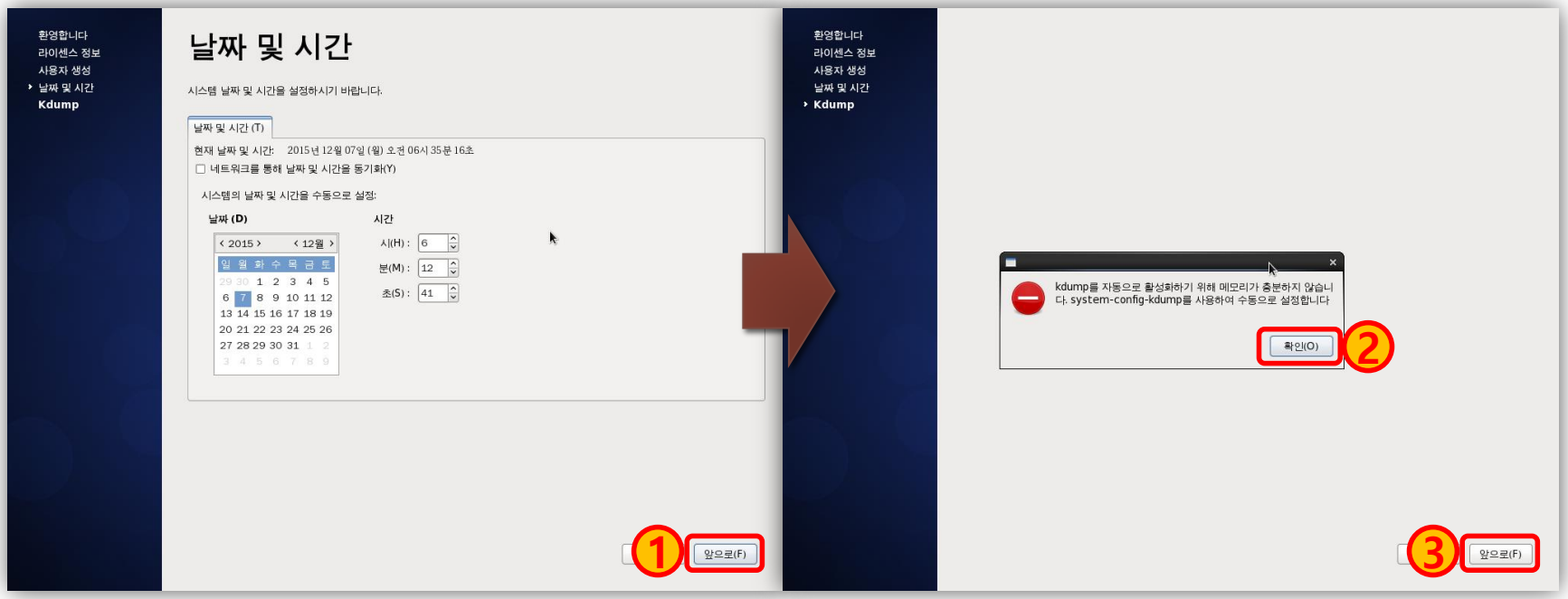
좌측 그림과 같이 CentOS 설치가 완료 되면 "재부팅(T)" 버튼을 클릭해 시스템을 재부팅 하고 우측 그림의 ②번과 같이 "예 라이선스 조항에 동의합니다(Y)"를 선택하고 "앞으로(F)" 버튼을 클릭해 다음 단계로 넘어간다.

Linux(Cent OS) 설치 – 일반 사용자 생성



좌측 그림과 같이 CentOS 일반 사용자 계정을 생성하는 대화상자가 나타나면 사용자 이름과 성명에 "centosuser"을 입력하고 암호는 "12345678"로 지정한 후 "앞으로(F)" 버튼을 클릭하게 되면 암호가 단순해 취약하다는 안내창이 화면에 뜨는데 여기서 ② 번 "예(Y)" 버튼을 클릭한 후 "앞으로(F)" 버튼을 클릭해 다음 단계로 넘어간다.

Linux(Cent OS) 설치 – 날짜 및 시간 설정



좌측 그림과 같이 날짜 및 시간 설정 화면에서 날짜를 설정하고 "앞으로(F)" 버튼을 클릭하면 우측 그림과 같이 메모리 부족으로 Kdump를 활성화 할 수 없다는 경고 창이 뜨는데 여기서 "확인(O)"를 클릭하고 "앞으로(F)" 버튼을 클릭하여 나타나는 화면에서 "완료(F)" 버튼을 클릭해 설정을 마무리 한다. 그러면 가상 시스템이 재부팅 되고 "centosuser" 로그인 화면이 나타난다.

Linux(Cent OS) 설치 – root 사용자로 로그인

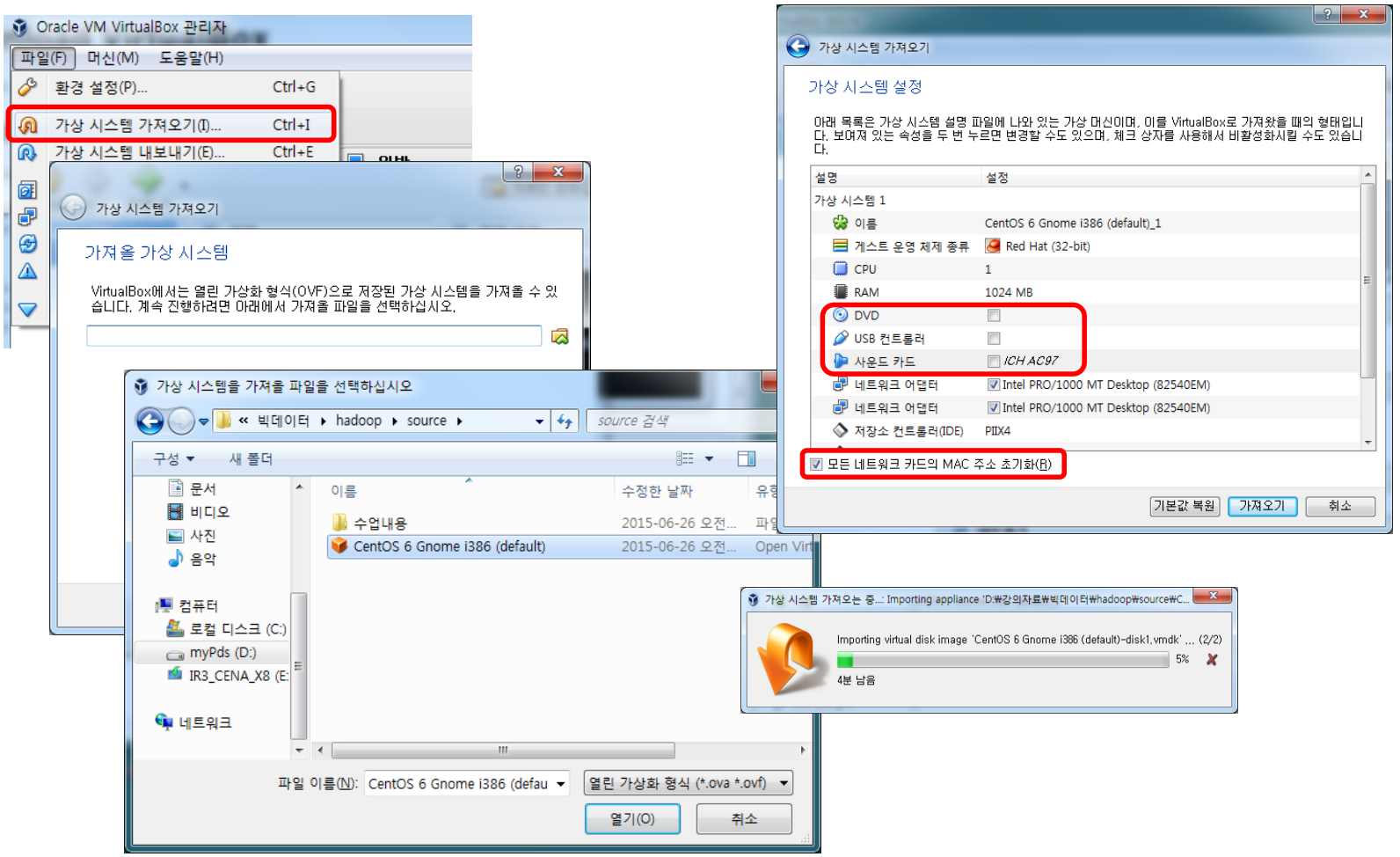
시스템이 재부팅 되면 좌측 그림과 "centosuser"로 로그인 화면이 나타난다. 우리는 슈퍼 유저인 root 사용자로 로그인 할 것 이므로 "기타..."를 클릭해서 우측 위쪽의 그림과 같이 root 사용자로 로그인 하자.
우측의 아래쪽 그림은 root 사용자로 로그인 했을 때 나타나는 경고 창으로 이 창에서 "이 메시지를 다시 보지 않습니다"를 선택한 상태에서 "닫기(C)" 버튼을 클릭하면 리눅스 GUI 환경인 X-Window이 나타날 것이다.

Linux(Cent OS) 기본 명령어

별도 제공되는 자료 참고

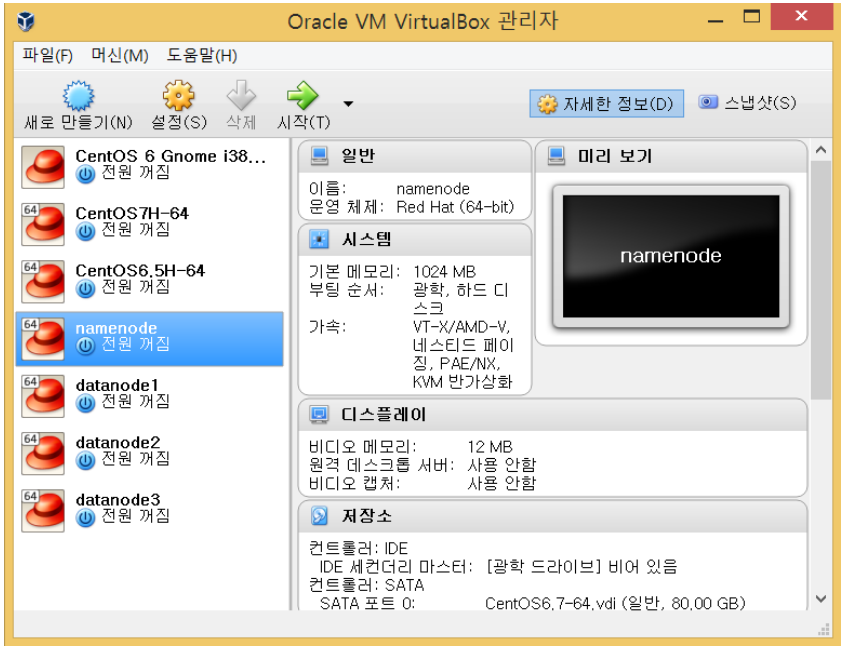
DataNode 구축(Virtual System Loading)

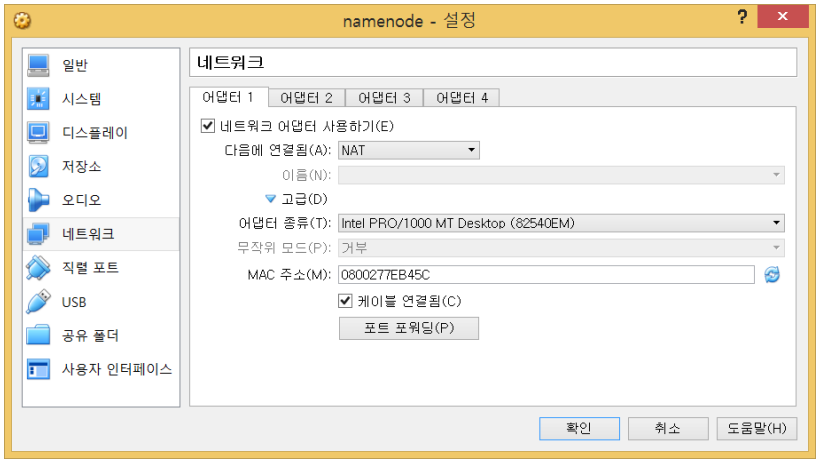
가상 시스템 가져오기로 가상머신을 만들 때

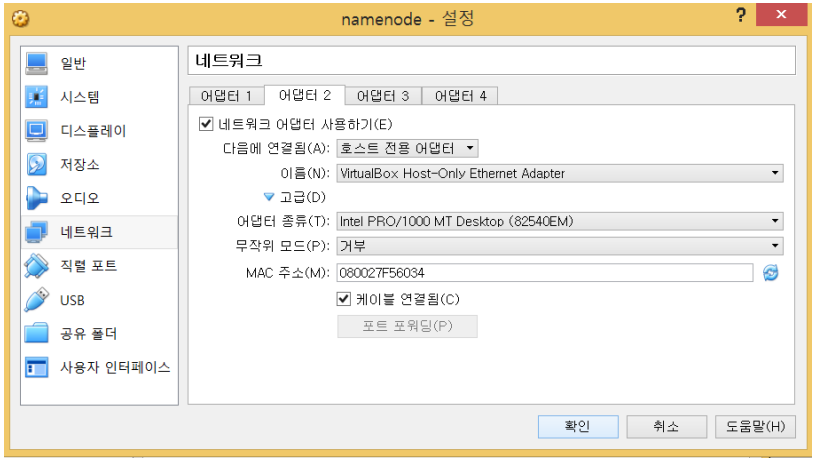


DataNode 구축(Virtual System Setting)

가상 시스템 가져오기로 가상 머신을 만드는 경우



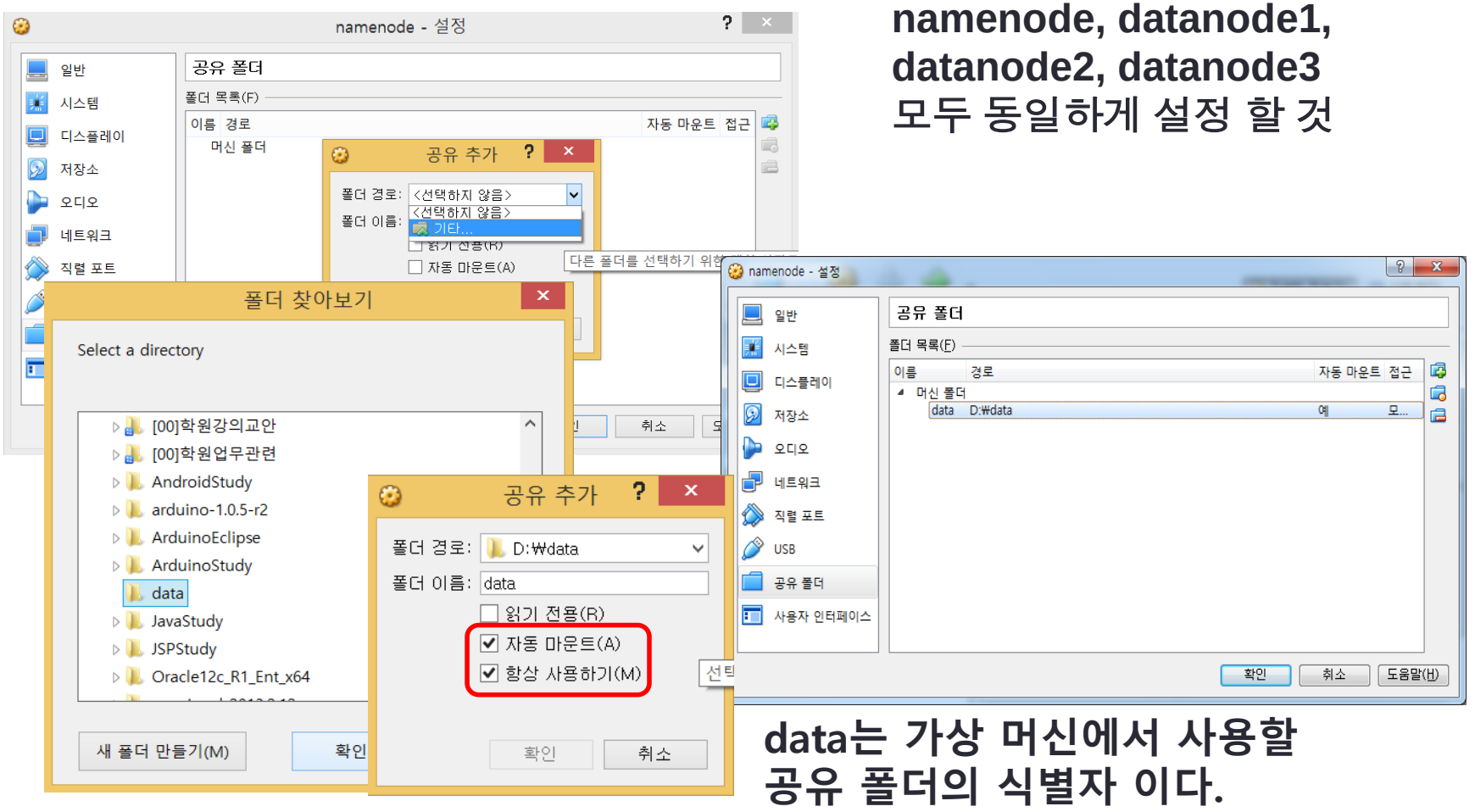




namenode, datanode1, datanode2, datanode3 모두 동일하게 설정 할 것

DataNode 구축(Vitual System Setting)

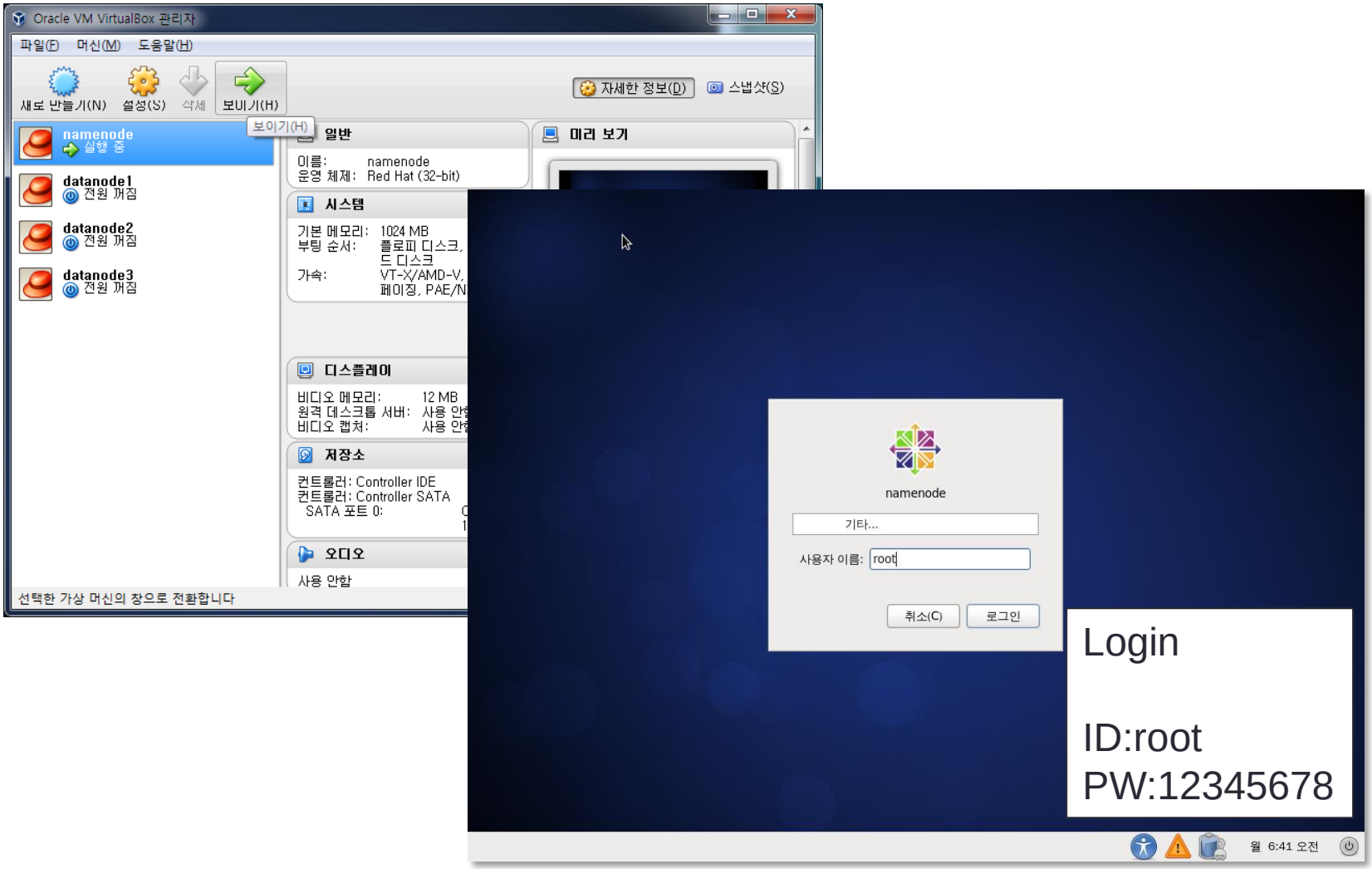
가상 시스템 가져오기로 가상 머신을 만드는 경우



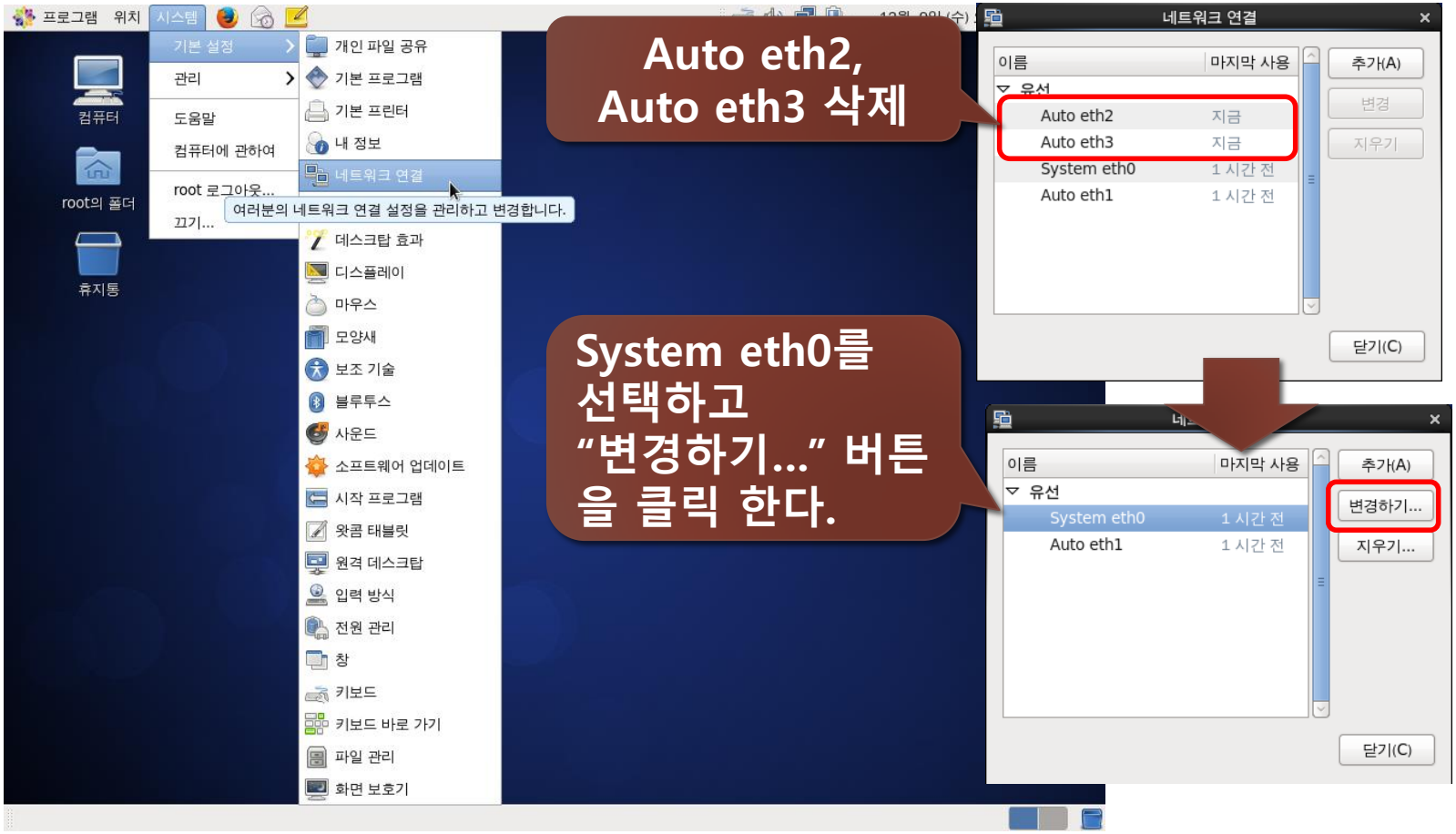
namenode, datanode1, datanode2, datanode3 모두 동일하게 설정 할 것

data는 가상 머신에서 사용할 공유 폴더의 식별자 이다.

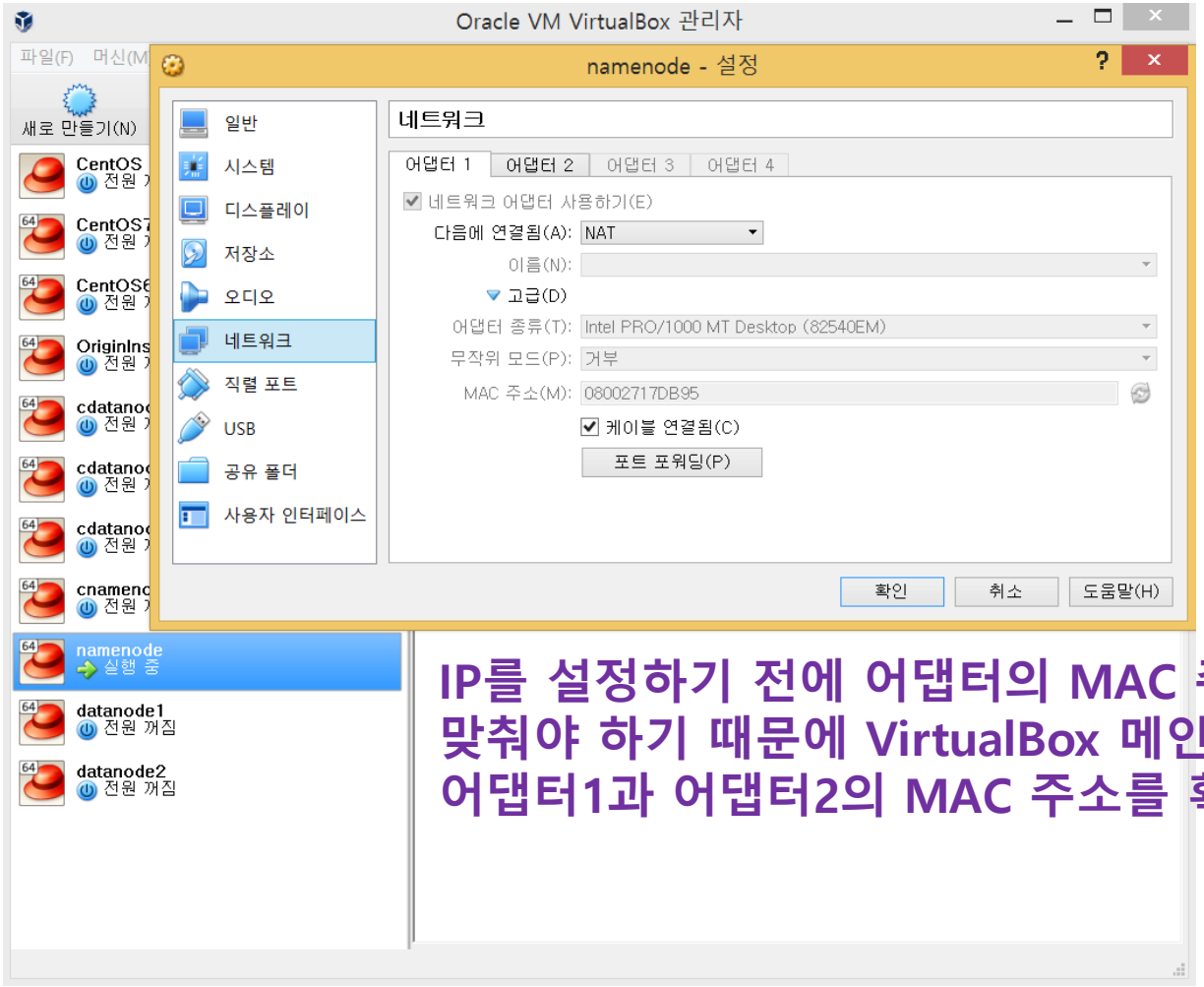
DataNode 구축 (namenode 실행)



DataNode 구축(네트워크 – IP 설정)



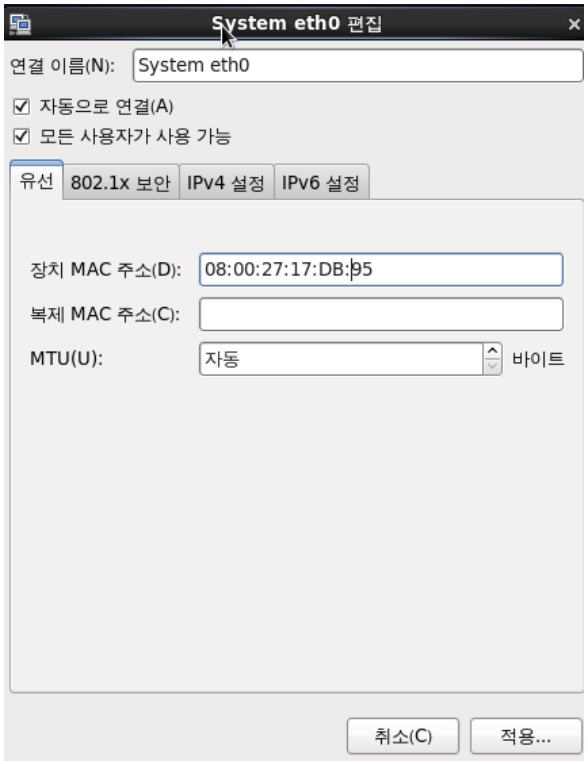
DataNode 구축(네트워크 – IP 설정)

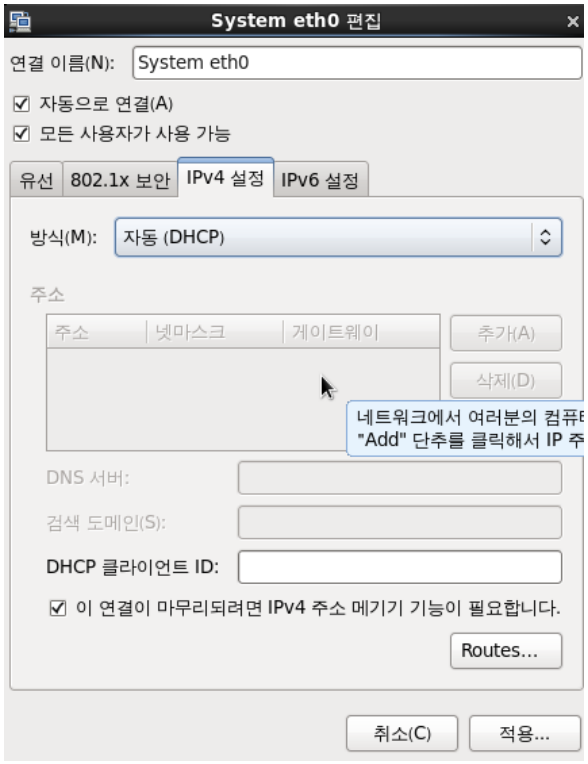


IP를 설정하기 전에 어댑터의 MAC 주소를 맞춰야 하기 때문에 VirtualBox 메인에서 어댑터1과 어댑터2의 MAC 주소를 확인한다.

DataNode 구축(네트워크 – IP 설정)

System eth0 장치

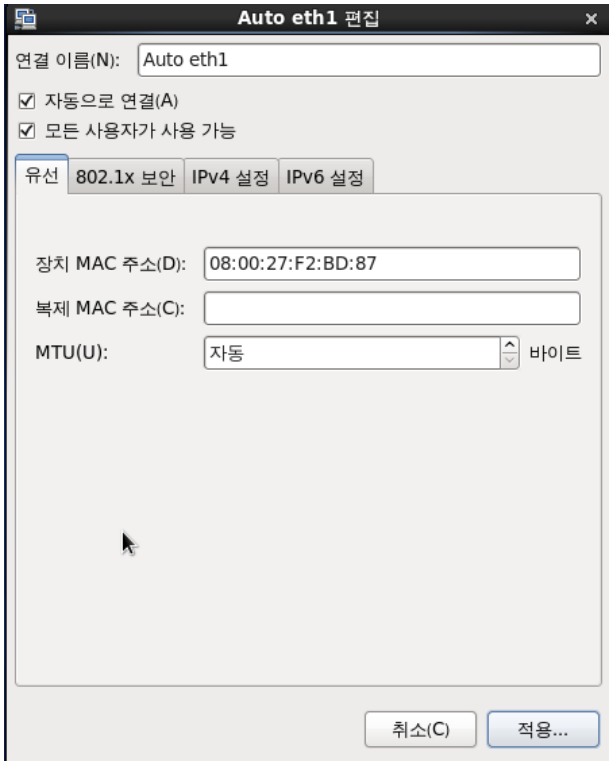


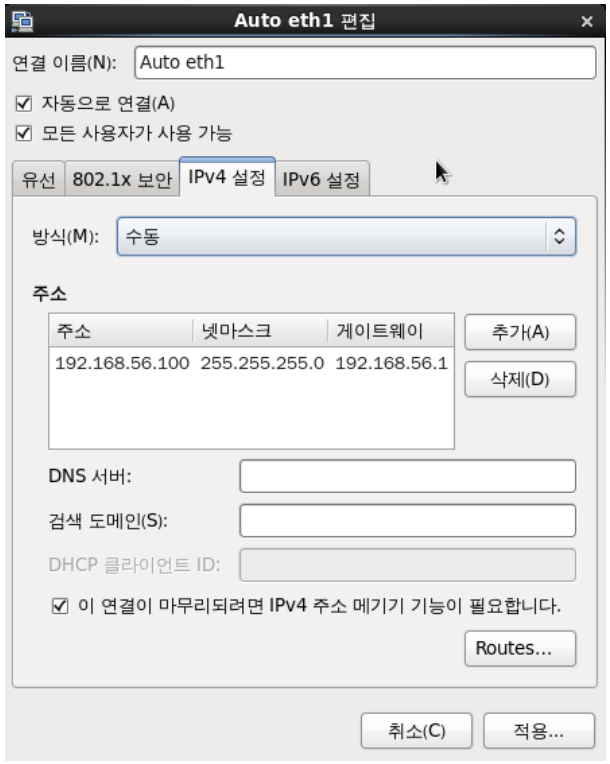


유선 탭의 장치 MAC 주소(D) 입력란에 앞에서 확인한 어댑터1의 MAC 주소를 입력하고 IPv4 설정 탭에서 “자동(DHCP)” 방식을 선택한다.

DataNode 구축(네트워크 – IP 설정)

Auto eth1 장치





유선 탭의 장치 MAC 주소(D) 입력란에 앞에서 확인한 어댑터2의 MAC 주소를 입력하고 IPv4 설정 탭에서 “수동” 방식을 선택한 후 우측 그림과 같이 IP와 넷마스크, 게이트웨이를 입력 한다. 수정된 IP를 확인하기 이전에 먼저 리부팅을 한다.

DataNode 구축(네트워크 – IP 설정)

나머지 datanode1, datanode2, datanode3도 아래를 참고해서 namenode를 설정한 것과 같이 Auto eth1 어댑터를 설정한다.

-. IP Address
namenode : 192.168.56.100
datanode1 : 192.168.56.101
datanode2 : 192.168.56.102
datanode3 : 192.168.56.103

-. NETMASK
255.255.255.0

-. GATEWAY
192.168.56.1

-. 네트워크 재 시작 명령
/etc/rc.d/init.d/network restart

이 명령으로 에러가 날 때는 ifconfig 명령으로 어댑터의 장치 명을 확인하고 /etc/sysconfig/network-scripts/ 폴더의 네트워크 설정 파일 ifcfg-eth0, ifcfg-Auto_eth1 등의 파일에서 DEVICE로 지정한 이름과 같은지를 확인해서 다르면 VI 에디터를 통해 수정해야 한다.

DataNode 구축(네트워크 – Host 설정)

- Host name 설정

  #vi /etc/sysconfig/network

  NETWORKING = yes

  HOSTNAME = namenode

datanode1, datanode2, datanode3 모두 HOSTNAME과 NETWORK 설정을 해줘야 함.

- Network 설정 추가

  #vi /etc/sysconfig/network-scripts/ifcfg-Auto_eth1

  NETMASK=255.255.255.0

  IPADDR=192.168.56.100

  GATEWAY=192.168.56.1

  - 서버 리부팅 또는 네트워크 재 시작 명령

    shutdown –r now 또는 /etc/rc.d/init.d/network restart

DataNode 구축(RSA 보안키 설정)

하둡의 네임서버가 데이터노드와 ssh(secure shell) 데이터 노드 가상 머신과 ssh 연결시 패스워드를 묻지 않도록 설정해 주어야 합니다.

ssh-keygen 후에 ~/.ssh/* 파일을 데이터 노드 가상 머신에 복사합니다.

- 공개키 만들기

```
# ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
Generating public/private rsa key pair.
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
c2:0a:31:38:f2:1a:fa:1a:25:74:e3:d2:ec:63:6d:eb root@nameno
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|  .              |
| +. oo           |
| oo=o. .         |
| o. ++   o S     |
| . =o. . .  .    |
| +   +. o        |
|  o.  o .        |
| ...  .E         |
+-----------------+
```

DataNode 구축(RSA 보안키 설정)

```
# cd .ssh
# cp id_rsa.pub authorized_keys
# ls
authorized_keys    id_rsa    id_rsa.pub    known_hosts
```

- 공개키 복사

```
# scp ./* 192.168.56.101:~/.ssh/
```

위의 명령으로 공개 키를 복사하려고 시도하면 하면 다음 페이지와 같이 192.168.56.101 IP를 가진 호스트의 신뢰성을 설정할 수 없는데 그래도 연결할 것인지를 물어 온다. 공개키가 각 datanode 들에게 제대로 복사되었다면 다음 부터는 root 계정의 암호 입력 없이 SSH로 바로 접속이 가능하다.

DataNode 구축(RSA 보안키 설정)

- 아래와 같이 “yes”와 root 계정 암호인 “12345678”를 입력한다.

```
The authenticity of host '192.168.56.101 (192.168.56.101)' can't be established.
RSA key fingerprint is fl:ba:20:cc:a0:4d:56:b0:37:8a:5f:aa:22:ed:1c:67.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.56.102' (RSA) to the list of known hosts.
root@192.168.56.102's password:12345678
```

- 공개키는 아래와 같이
  datanode1, datanode2, datanode3 모두에 동일하게 복사

```
# scp ./* 192.168.56.101:~/.ssh/
# scp ./* 192.168.56.102:~/.ssh/
# scp ./* 192.168.56.103:~/.ssh/
```

DataNode 구축(Host 및 방화벽 설정)

- 노드별 호스트명을 /etc/hosts 파일에 작성하고 각 datanode에 복사
  - 192.168.56.100 namenode
  - 192.168.56.101 datanode1
  - 192.168.56.102 datanode2
  - 192.168.56.103 datanode3
  - # scp /etc/hosts datanode1:/etc/
  - # scp /etc/hosts datanode2:/etc/
  - # scp /etc/hosts datanode3:/etc/

- 방화벽 설정
  - # service iptables stop
  - # ssh datanode1 service iptables stop
  - # ssh datanode2 service iptables stop
  - # ssh datanode3 service iptables stop

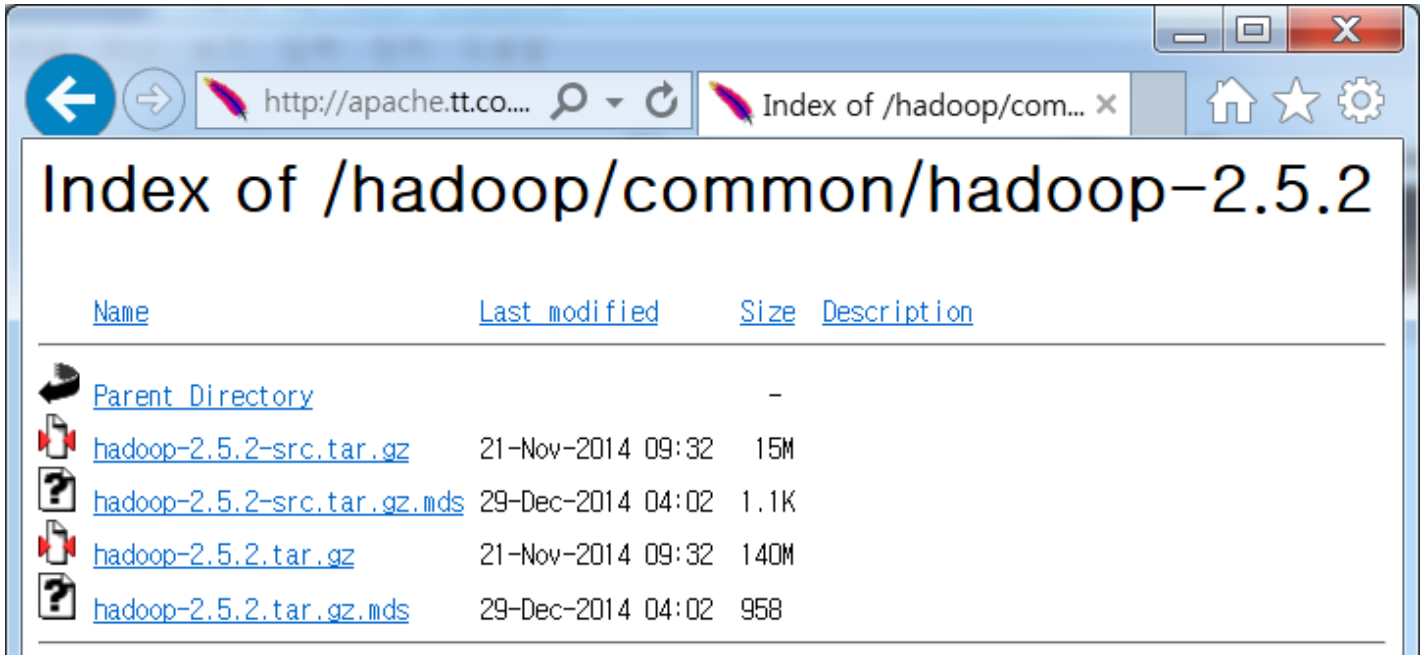
Chapter 04 Hadoop 2.0 설치 및 배포

- Hadoop 2.0 다운로드 및 설치
- JDK 다운로드 및 설치
- JDK와 Hadoop 환경변수 설정
- Hadoop 환경설정
- 배포하기 및 실행
- Hadoop 실행하기

Hadoop 2.0 다운로드

- **hadoop 다운로드**

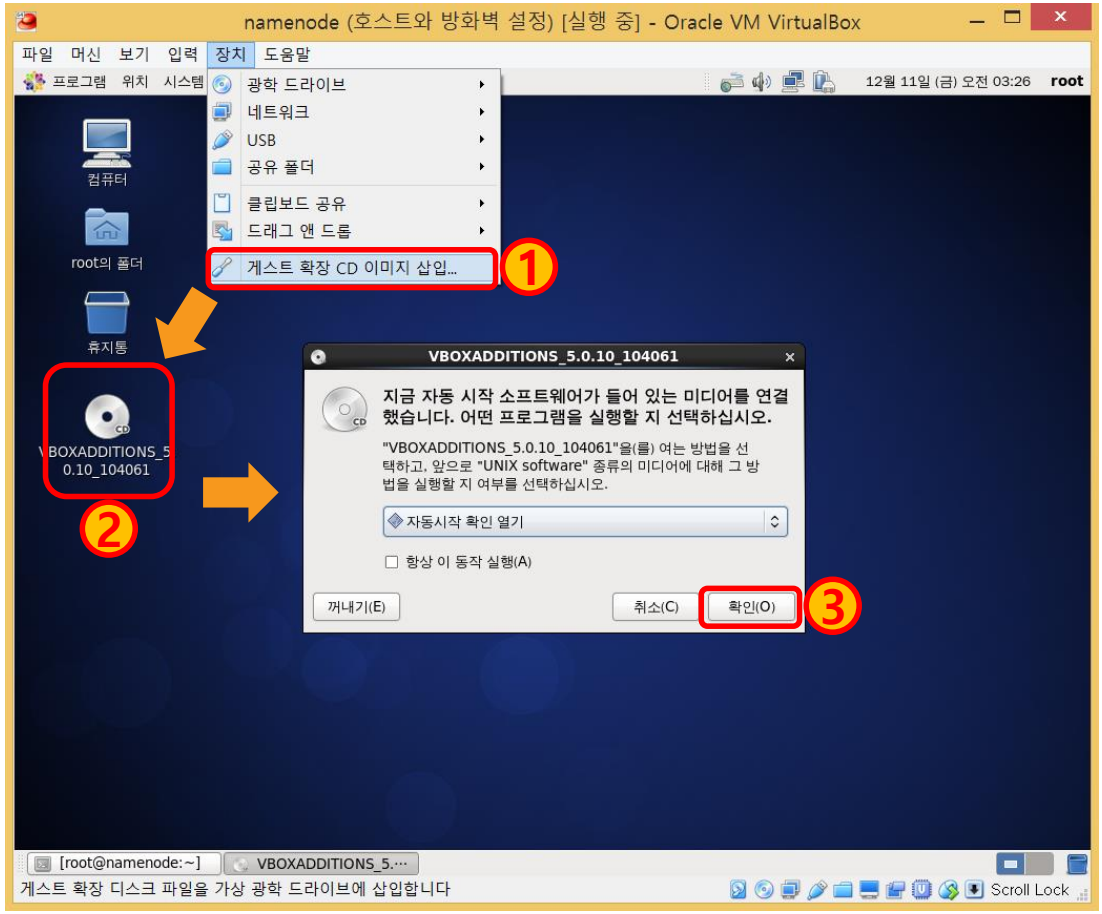
  http://apache.tt.co.kr/hadoop/common/hadoop-2.5.2/



http://apache.tt.co.kr/hadoop/common/hadoop-2.5.2/hadoop-2.5.2.tar.gz
관련 파일은 hadoop 배포사이트에서 직접 다운로드 가능함

Hadoop 설치하기

호스트 컴퓨터의 D:₩data₩를 공유 폴더로 사용하기 위해 마운트가 필요하다. 아래와 그림과 같이 "게스트 확장 프로그램"을 설치하자.



2 번에서 자동으로 3 번 화면이 나타나지 않으면 2 번 아이콘에 마우스 우 클릭하여 "자동시작 확인 열기"를 선택하면 된다.

Hadoop 설치하기

- 다운로드 받은 “hadoop-2.5.2.tar.gz” 파일을 D:₩data₩로 복사

```
# mkdir /home/root
# cd /home/root
# mkdir data
# mount -t vboxsf data  data
# ls data
# cp data/hadoop-2.5.2.tar.gz /home/root
# tar xvfz hadoop-2.5.2.tar.gz
# chown –R root:root hadoop-2.5.2/   <- 디렉토리의 소유권 설정
```

JDK 다운로드

http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html#jdk-7u80-oth-JPR

Java SE Development Kit 7u80

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux x86	130.44 MB	jdk-7u80-linux-i586.rpm
Linux x86	147.68 MB	jdk-7u80-linux-i586.tar.gz
Linux x64	131.69 MB	jdk-7u80-linux-x64.rpm
Linux x64	146.42 MB	jdk-7u80-linux-x64.tar.gz
Mac OS X x64	196.94 MB	jdk-7u80-macosx-x64.dmg
Solaris x86 (SVR4 package)	140.77 MB	jdk-7u80-solaris-i586.tar.Z
Solaris x86	96.41 MB	jdk-7u80-solaris-i586.tar.gz
Solaris x64 (SVR4 package)	24.72 MB	jdk-7u80-solaris-x64.tar.Z
Solaris x64	16.38 MB	jdk-7u80-solaris-x64.tar.gz
Solaris SPARC (SVR4 package)	140.03 MB	jdk-7u80-solaris-sparc.tar.Z
Solaris SPARC	99.47 MB	jdk-7u80-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	24.05 MB	jdk-7u80-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	18.41 MB	jdk-7u80-solaris-sparcv9.tar.gz
Windows x86	138.35 MB	jdk-7u80-windows-i586.exe
Windows x64	140.09 MB	jdk-7u80-windows-x64.exe

Back to top

JDK 설치하기

호스트 컴퓨터의 공유 폴더인 D:₩data에 다운로드 받은 jdk-7u80-linux-x64.tar.gz 파일을 복사하고 namenode에서 아래와 같이 작업한다.

```
# mount -t vboxsf data  data
# cp data/ jdk-7u80-linux-x64.tar.gz /home/root/
# tar xvfz jdk-7u80-linux-x64.tar.gz
# chown -R root:root jdk1.7.0_80/ <- 디렉토리의 소유권 설정
```

JDK 경로를 쉽게 접근하기 위해 java7 이라는 이름으로 심볼릭 링크 설정
(윈도우에서 바로가기와 비슷한 기능)

```
# ln -s jdk1.7.0_80 java7
# ls –l 또는 ll 명령으로 확인
```

JDK와 하둡 환경변수 설정(1)

vi /etc/profile

위의 명령으로 profile 파일을 열어 pathmunge() 함수 아래에 추가

```
export JAVA_HOME=/home/root/java7
export HADOOP_HOME=/home/root/hadoop-2.5.2
export PATH=$PATH:$JAVA_HOME/bin
export CLASSPATH=$JAVA_HOME/jre/lib/ext:$JAVA_HOME/lib/toos.jar
pathmunge $HADOOP_HOME/bin
pathmunge $HADOOP_HOME/sbin
```

아래 명령으로 수정된 profile 파일의 내용을 시스템에 적용한다.
터미널 마다 모두 해줘야 함 – 리부팅 하면 모든 터미널에 적용된다.

source /etc/profile

JDK와 하둡 환경변수 설정(2)

- **hadoopo path 확인**

```
# hadoop version
Hadoop 2.5.2
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r cc72e9b000545b86b75a61f4835eb86d57bfafc0
Compiled by jenkins on 2014-11-14T23:45Z
Compiled with protoc 2.5.0
From source with checksum df7537a4faa4658983d397abf4514320
This command was run using /home/root/hadoop-2.5.2/share/hadoop/common/hadoop-common-2.5.2.jar
```

JDK와 하둡 환경변수 설정(3)

- **java path 확인**

```
# java -version
java version "1.7.0_79"
OpenJDK Runtime Environment (rhel-2.5.5.4.el6-x86_64 u79-b14)
OpenJDK 64-Bit Server VM (build 24.79-b02, mixed mode)
```

CentOS 6.7에서 java –version 명령은 디폴트로 설치되는 java를 찾아 실행하기 때문에 우리가 설치한 자바 버전이 아닌 CentOS6.7을 설치할 때 기본으로 설치되는 OpenJDK 버전이 화면에 출력된다.

JDK와 하둡 환경변수 설정(4)

아래와 같이 update-alternatives java 명령으로 확인해 보면 OpenJDK 1.7.0이 기본으로 설정되어 있는 것을 확인할 수 있다.

```
# update-alternatives --config java

2 개의 프로그램이 'java'를 제공합니다.

  선택    명령
-----------------------------------------------
*+ 1           /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/bin/java
   2           /usr/lib/jvm/jre-1.6.0-openjdk.x86_64/bin/java

현재 선택[+]을 유지하려면 엔터키를 누르고, 아니면 선택 번호를 입력하십시오:enter
```

JDK와 하둡 환경변수 설정(5)

아래와 같이 update-alternatives --install 명령으로 /usr/bin/java 에 우리가 설치한 JDK 링크를 추가하고 updata-alternatives –config java 명령을 사용해 우리가 설치한 JDK를 지정하면 된다.

update-alternatives --install <링크> <이름> <경로> <우선순위>

```
# update-alternatives --install /usr/bin/java java /home/root/java7/bin/java 3
# update-alternatives --config java

3 개의 프로그램이 'java'를 제공합니다.
  선택    명령
-----------------------------------------------
*+ 1           /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/bin/java
   2           /usr/lib/jvm/jre-1.6.0-openjdk.x86_64/bin/java
   3           /home/root/java7/bin/java
현재 선택[+]을 유지하려면 엔터키를 누르고, 아니면 선택 번호를 입력하십시오:3
```

JDK와 하둡 환경변수 설정(6)

다시 아래와 같이 update-alternatives –config java 명령으로 확인해 보면 우리가 설치한 JDK가 기본으로 설정되어 있는 것을 확인할 수 있을 것이다.

```
# update-alternatives --config java

3 개의 프로그램이 'java'를 제공합니다.
  선택    명령
-----------------------------------------------
*   1           /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/bin/java
    2           /usr/lib/jvm/jre-1.6.0-openjdk.x86_64/bin/java
 + 3           /home/root/java7/bin/java
현재 선택[+]을 유지하려면 엔터키를 누르고, 아니면 선택 번호를 입력하십시오:enter
```

JDK와 하둡 환경변수 설정(7)

- **java path 확인**

```
# java -version
java version "1.7.0_80"
Java(TM) SE Runtime Environment (build 1.7.0_80-b15)
Java HotSpot(TM) 64-Bit Server VM (build 24.80-b11, mixed mode)
```

현재 시스템에서 사용하는 기본 JDK가 CentOS 6.7을 설치 할 때 기본으로 설치된 OpenJDK 버전에서 우리가 설치한 JDK 1.7.0u80으로 바뀐 것을 확인할 수 있다.

Hadoop 환경 설정(1)

- 하둡 설정파일 종류

  분산 하둡 클러스터를 구축하기 위해 아래의 파일에 환경 정보를 설정

  하둡의 설정파일 위치

  - 1.x : $HADOOP_HOME/conf
  - 2.x : $HADOOP_HOME/etc/hadoop

  1. 하둡 실행을 위한 환경 설정 파일
     - ✓ hadoop-env.sh
  2. 하둡 분산 파일 시스템과 하둡 맵리듀스를 동작시키기 위한 파일
     - ✓ core-site.xml
     - ✓ hdfs-site.xml
     - ✓ mapred-site.xml
  3. 각 노드들의 동작에 필요한 파일
     - ✓ masters
     - ✓ slaves

Hadoop 환경 설정(2)

- hadoop-env.sh 설정

  하둡 관련 프로세스들이 동일한 환경으로 동작하기 위한 설정으로
  분산 구축을 위해 JDK 경로, 클래스 패스, 데몬 실행 옵션 등을 설정

  # cd $HADOOP_HOME/etc/hadoop

  # ls

  # vi hadoop-env.sh

```
# The only required environment variable is JAVA_HOME.  All others are
# optional.  When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
export JAVA_HOME=/home/root/java7

# The jsvc implementation to use. Jsvc is required to run secure datanodes.
#export JSVC_HOME=${JSVC_HOME}
```

Hadoop 환경 설정(3)

- core-site.xml 설정
  하둡 분산 파일 시스템(HDFS)과 맵리듀스에 공통으로 사용할 환경 설정

```
# vi core-site.xml
<configuration>
        <property>
                <name>fs.default.name</name>
                <value>hdfs://namenode:9000</value>
        </property>
        <property>
                <name>hadoop.tmp.dir</name>
                <value>/home/root/hadoop-2.5.2/tmp</value>
        </property>
</configuration>
```

Hadoop 환경 설정(4)

- **hdfs-site.xml 설정**

  하둡 분산 파일 시스템(HDFS)의 실행 환경을 설정한다. 현재 구성에서는 분산 파일 시스템의 서버(datanode)가 세 개이므로 복제 개수(Replication Count)를 3개로 지정하였다.

  #vi hdfs-site.xml

```
<configuration>
        <property>
                <name>dfs.replication</name>
                <value>3</value>
        </property>
</configuration>
```

Hadoop 환경 설정(5)

- **slaves**

  데이터노드와 테스크 트래커가 동작할 수 있도록 설정하는 파일.

  #vi slaves
  datanode1
  datanode2
  datanode3

  slaves의 기존 내용 localhost는 삭제한다.

JDK 배포하기(1)

각 datanode: /home 디렉터리에 root 디렉터리를 생성하고 JDK 배포

```
# ssh datanode1
# mkdir /home/root
# cd /home/root/
# scp -r jdk1.7.0_80 datanode1:/home/root/
# scp -r jdk1.7.0_80 datanode2:/home/root/
# scp -r jdk1.7.0_80 datanode3:/home/root/
```

jdk 배포

각 노드에 접속해 JDK 경로를 쉽게 접근하기 위해 java7 이라는 이름으로 심볼릭 링크 설정(윈도우에서 바로가기와 비슷한 기능)

```
# ssh datanode1
# cd /home/root/
# ln -s jdk1.7.0_80 java7
# ls -l 또는 ll 명령으로 확인
```

JDK 배포하기(2)

각 datanode도 update-alternatives --install 명령으로 /usr/bin/java 에 각 노드에 배포한 JDK 링크를 추가하고 updata-alternatives –config java 명령을 사용해 우리가 설치한 JDK를 지정해 줘야 한다.

```
update-alternatives --install <링크> <이름> <경로> <우선순위>
# ssh datanode1
# update-alternatives --install /usr/bin/java java /home/root/java7/bin/java 3
# update-alternatives --config java

3 개의 프로그램이 'java'를 제공합니다.
  선택     명령
-----------------------------------------------
*+ 1           /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/bin/java
   2           /usr/lib/jvm/jre-1.6.0-openjdk.x86_64/bin/java
   3           /home/root/java7/bin/java
현재 선택[+]을 유지하려면 엔터키를 누르고, 아니면 선택 번호를 입력하십시오:3
```

Hadoop 배포하기

```
# cd /home/root/

# scp /etc/profile datanode1:/etc/
# scp /etc/profile datanode2:/etc/
# scp /etc/profile datanode3:/etc/
```

profile 배포

각 datanode: /home 디렉터리에 root 디렉터리를 생성하고 Hadoop 배포

```
# ssh datanode1
# mkdir /home/root

# scp -r hadoop-2.5.2 datanode1:/home/root/
# scp -r hadoop-2.5.2 datanode2:/home/root/
# scp -r hadoop-2.5.2 datanode3:/home/root/
```

hadoop 배포

하둡 2.0 네임노드 포멧

```
# hdfs namenode -format
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = namenode/192.168.56.100
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.5.2
STARTUP_MSG:   classpath = .....
STARTUP_MSG:   java = 1.7.0_80 ....
15/12/14 01:12:37 INFO common.Storage: Storage directory /home/root/hadoop-2.5.2/tmp/dfs/name has been successfully formatted.  .....
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at namenode/192.168.56.100
************************************************************/
```

서버의 ip와 host 네임확인

하둡 2.0 실행(1)

 # cd $HADOOP_HOME/sbin PATH 설정으로 어디서든 실행 가능

 # ./start-dfs.sh (종료 ./stop-dfs.sh)

.....

Starting namenodes on [namenode]

namenode: namenode running as process 3763. Stop it first.

datanode2: starting datanode, logging to /home/root/hadoop-2.5.2/logs/hadoop-root-datanode-datanode2.out

datanode3: starting datanode, logging to /home/root/hadoop-2.5.2/logs/hadoop-root-datanode-datanode3.out

datanode1: starting datanode, logging to /home/root/hadoop-2.5.2/logs/hadoop-root-datanode-datanode1.out

Starting secondary namenodes [0.0.0.0]

0.0.0.0: reverse mapping checking getaddrinfo for localhost [127.0.0.1] failed - POSSIBLE BREAK-IN ATTEMPT!

0.0.0.0: secondarynamenode running as process 3938. Stop it first.

.....

하둡 2.0 실행(2)

```
# ssh datanode1
# jps <- 자바 PID(Process ID) 리스트
3572 Jps
3495 DataNode
```

namenode 브라우저를 열어서 http://namenode:50070 연결하면 namenode information화면이 나온다.

core-site.xml에 hadoop.tmp.dir을 /home/root/hadoop-2.5.2/tmp 로 지정했기 때문에 하둡이 실행되면 디렉터리 구성은 다음과 같다.

- Namenode 파일시스템 이미지(namenode와 각 datanode에 생성) /home/root/hadoop-2.5.2/tmp/dfs/name 디렉터리에 저장

하둡 2.0 실행(3)

- 보조네임노드 파일시스템 이미지(namenode와 각 datanode에 생성)
  /home/root/hadoop-2.5.2/tmp/dfs/namesecondary

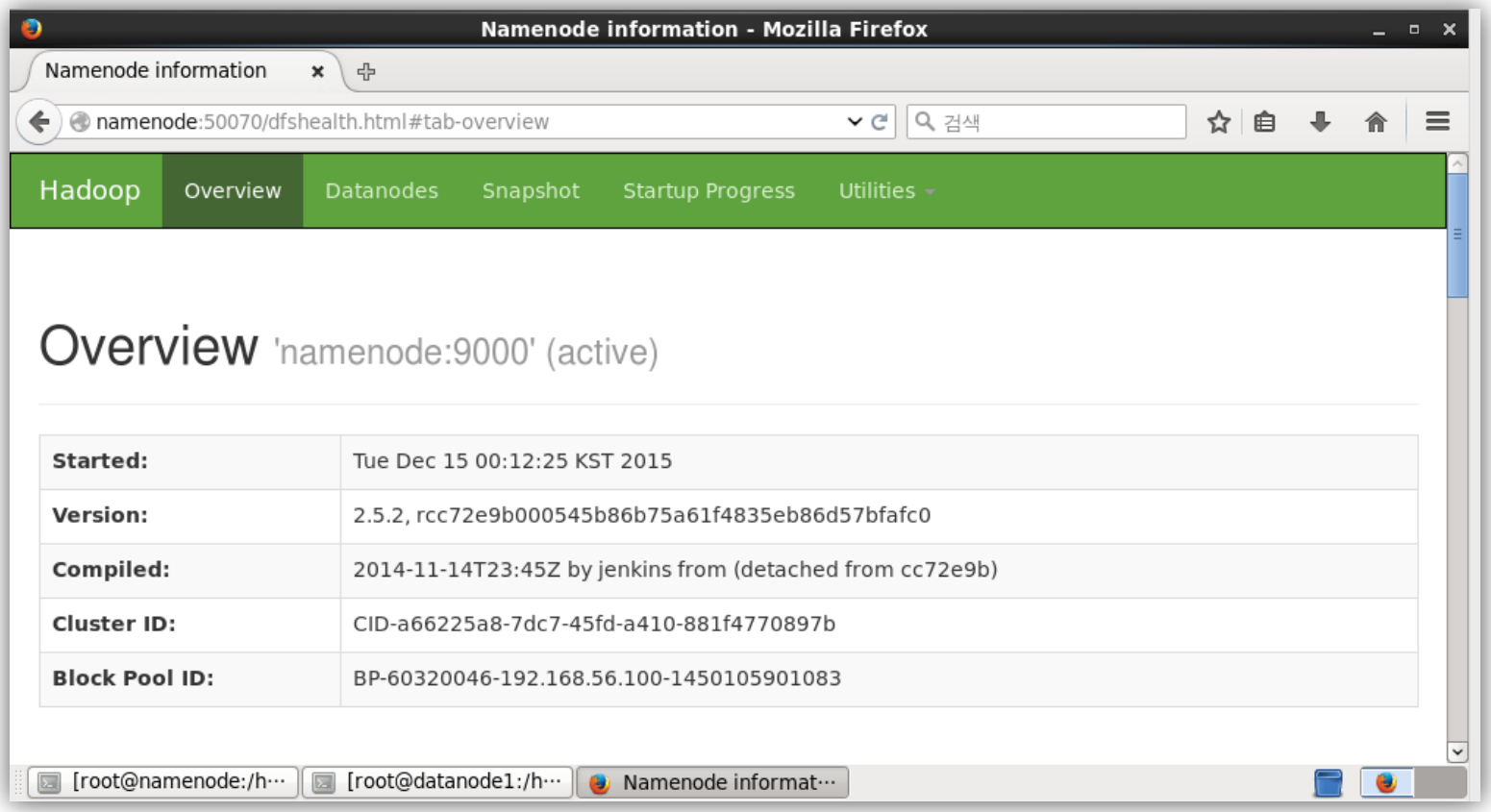
- 로그 디렉터리(namenode와 각 datanode에 생성)
  $HADOOP_HOME/logs/

- 데이터 저장(각 datanode에 생성)
  /home/root/hadoop-2.5.2/tmp/dfs/data

  namenode를 포맷한 후에 재 포맷했을 때 datanode가 실행되지 않는 경우가 종종 발생하는데 이때 hadoop-사용자-노드명-호스트명.log를 확인해 보면 namenode의 namespaceID와 datanode의 namespaceID가 일치하지 않아서 발생하는 현상으로 HDFS의 data 폴더를 삭제하고 다시 생성한 후 namenode를 포맷하면 해결 할 수 있다.

하둡 2.0 실행(4)

- 브라우저 초기화면 (namenode:50070/dfshealth.html)

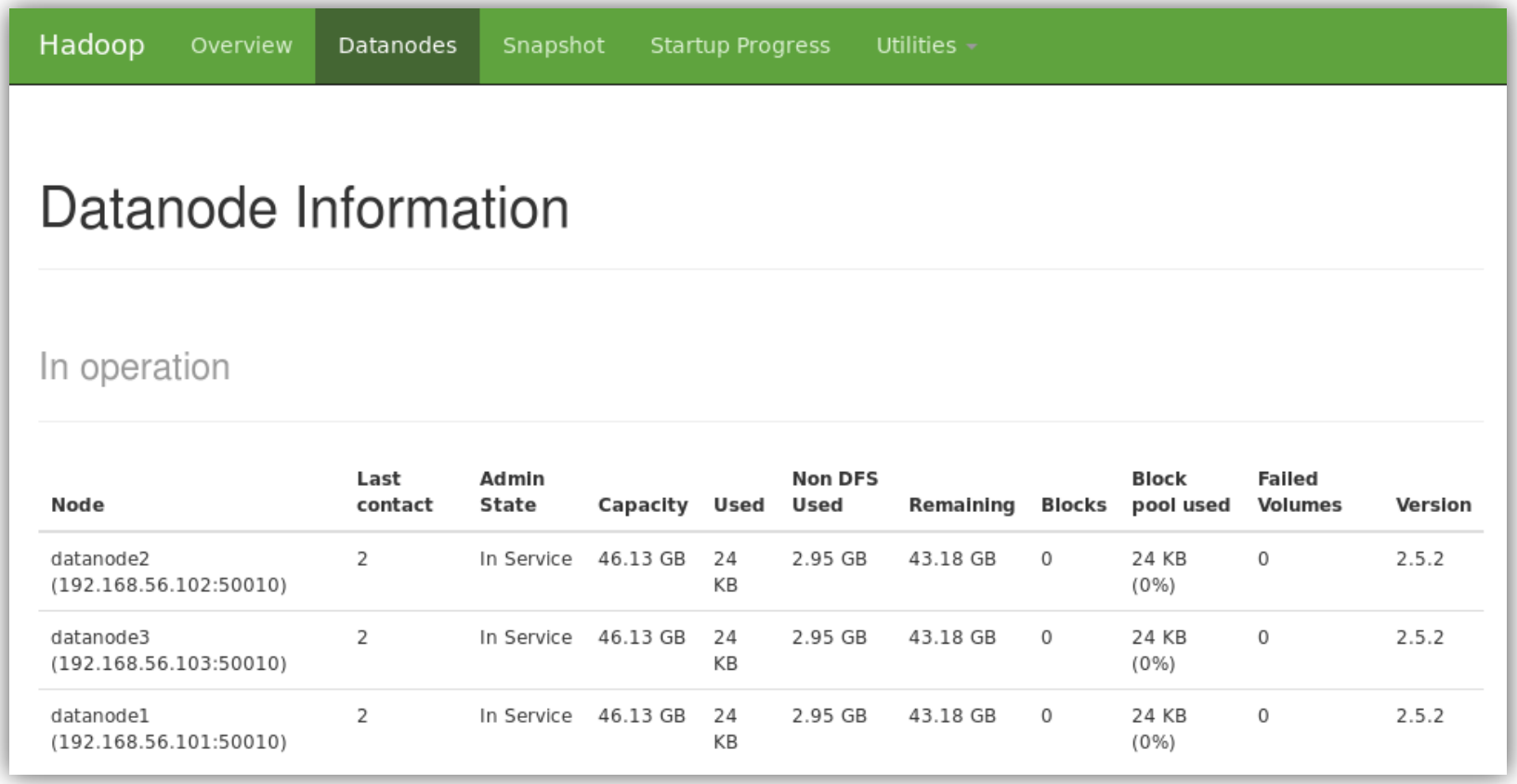


하둡 2.0 실행(5)

- 브라우저 초기화면 (namenode:50070/dfshealth.html)

Configured Capacity:	138.4 GB
DFS Used:	72 KB
Non DFS Used:	8.85 GB
DFS Remaining:	129.55 GB
DFS Used%:	0%
DFS Remaining%:	93.6%
Block Pool Used:	72 KB
Block Pool Used%:	0%
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	3 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Number of Under-Replicated Blocks	0
Number of Blocks Pending Deletion	0

하둡 2.0 실행(5)


Hadoop
Overview
Datanodes
Snapshot
Startup Progress
Utilities
Datanode Information
In operation
Node
Last contact
Admin State
Capacity
Used
Non DFS Used
Remaining
Blocks
Block pool used
Failed Volumes
Version
datanode2 (192.168.56.102:50010)
2
In Service
46.13 GB
24 KB
2.95 GB
43.18 GB
0
24 KB (0%)
0
2.5.2
datanode3 (192.168.56.103:50010)
2
In Service
46.13 GB
24 KB
2.95 GB
43.18 GB
0
24 KB (0%)
0
2.5.2
datanode1 (192.168.56.101:50010)
2
In Service
46.13 GB
24 KB
2.95 GB
43.18 GB
0
24 KB (0%)
0
2.5.2

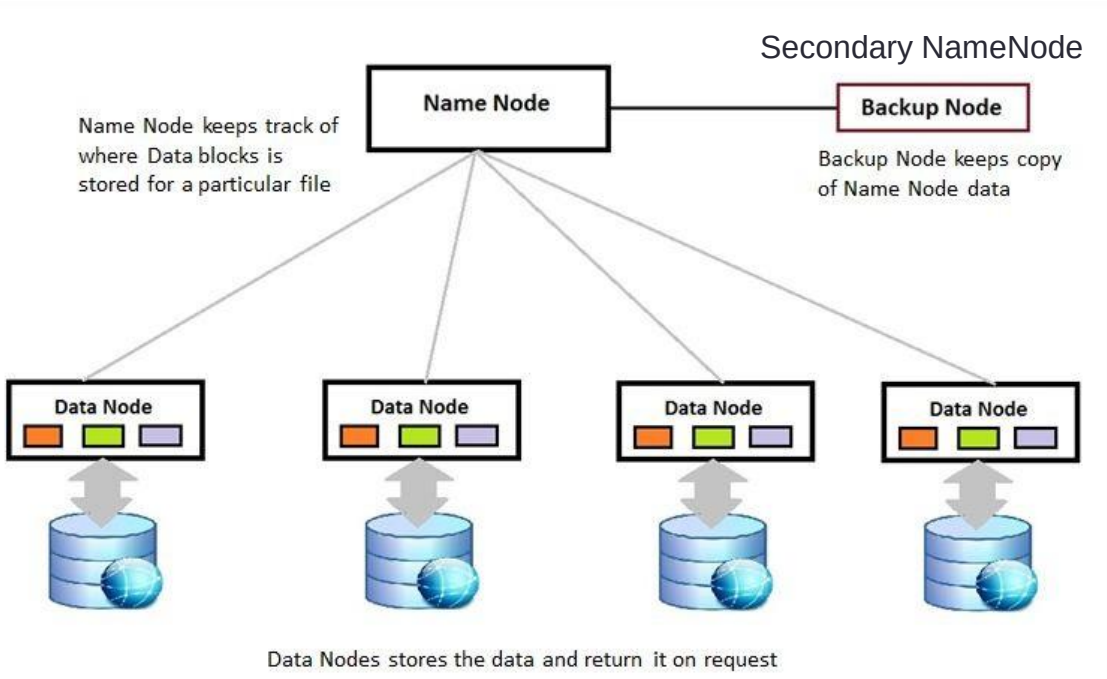
Chapter 05 Hadoop 분산 클러스터 구축

- 하둡 2.0 분산 클러스터 구성
- 하둡 2.0 관리
- 하둡의 데이터 처리 기술

하둡 Full Distributed 클러스터

- 하둡의 모든 기능이 갖추어진 클러스터
  - ✓ Master – Namenode와 JobTracker
  - ✓ Backup – SNN (Secondary Namenode)
  - ✓ Slave – DataNode와 TaskTracker

- 사용 목적
  - ✓ 분산저장
  - ✓ 분산연산



하둡 클러스터 구성

- **hadoop-env.sh 파일설정**

```
#cd $HADOOP_HOME/etc/hadoop
#ls
#vi hadoop-env.sh ->Java 홈 설정 변경
export JAVA_HOME=/usr/java/jdk1.6.0_35
```

✓ 하둡 실행 시, 아래와 같이 경고메시지가 계속 뜰 수 있다.
"Warning : $HADOOP_HOME is deprecated"

이럴 경우 아래 내용을 hadoop-env.sh 에 추가

```
export HADOOP_HOME_WARN_SUPPRESS=1
```

하둡 클러스터 구성

- core-site.xml 파일 수정

```
#vi core-site.xml
<configuration>
        <property>
                <name>fs.default.name</name>
                <value>hdfs://namenode:9000</value>
        </property>
        <property>
                <name>hadoop.tmp.dir</name>
                <value>/home/root/hadoop-2.5.2/tmp</value>
        </property>
</configuration>
```

하둡 클러스터 구성

- hdfs-site.xml 파일 수정

```
#vi hdfs-site.xml
<configuration>
        <property>
                <name>dfs.replication</name>
                <value>3</value>
        </property>
</configuration>
```

하둡 클러스터 구성

- **mapred-site.xml 파일 수정**

이 파일에서는 하둡 맵리듀스 관련 설정이 가능하다. 분산 구축을 위하여 Job tracker가 동작하는 환경을 설정해 준다. 여기서 mapreduce.framework.name 속성을 설정하여 어떤 방식으로 mapreduce job을 수행시킬 것인지 결정한다. Local/classic/yarn 중 선택하여 설정할 수 있다. local 이 기본 값이지만, 하둡 2.0버전에서 수정된 mapreduce Job을 실행시키기 위해서는 yarn으로 설정해야 한다.

```
# cp mapred-site.xml.template mapred-site.xml
# vi mapred-site.xml
<configuration>
        <property>
                <name>mapreduce.framework.name</name>
                <value>yarn</value>
        </property>
</configuration>
```

하둡 클러스터 구성

- yarn-site.xml 파일 수정

```
#vi yarn-site.xml
<configuration>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
    <property>
        <name>yarn.nodemanager.aux-
services.mapreduce_shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
    </property>
    <property>
        <name>yarn.resourcemanager.hostname</name>
        <value>namenode</value>
     </property>
</configuration>
```

하둡 클러스터 구성

- Hadoop 설정파일 배포

```
#cd $HADOOP_HOME/etc/hadoop
#scp ./* datanode1:$HADOOP_HOME/etc/hadoop
#scp ./* datanode2:$HADOOP_HOME/etc/hadoop
#scp ./* datanode3:$HADOOP_HOME/etc/hadoop
```

하둡 클러스터구성

- **Namenode 초기화**

#hdfs namenode –format

※ 에러메세지가 있다면 환경설정이 잘못된 것으로 확인하고 다시 실행 한다.

- 주의 : 데이터노드에 이미 데이터 파일이 생성된 상황에서 초기화하면 데이터 노드와 동기화 되지 않음.
- Java.io.IOException:Incompatible namespaceIDs
- 이 예외의 발생 원인은 namenode의 namespaceID와 datanode의 namespaceID가 일치하지 않아서 발생하는 것으로 namenode를 재 포맷하는 경우 datanode가 실행되지 않는 현상이 발생
- 각 datanode에서 rm –rf $HADOOP_HOME/tmp/dfs/data/* 으로 삭제 후 namenode를 포맷하고 하둡을 다시 실행

하둡 클러스터구성

- 실행 방법

start-dfs.sh
HDFS 시스템을 시작합니다. (stop-dfs.sh)

start-yarn.sh
yarn을 시작하여 resource manager와 node manager를 시작합니다.
(stop-yarn.sh)

#mr-jobhistory-daemon.sh start historyserver
history server 를 시작(mr-jobhistory-daemon.sh stop historyserver)

※ 브라우저에서
- HDFS 상태 확인 : http://namenode:50070/dfshealth.html
- Yarn 상태 확인 : http://namenode:8088

하둡 2.0 관리

- HDFS 정보

Configured Capacity:	138.4 GB
DFS Used:	72 KB
Non DFS Used:	8.86 GB
DFS Remaining:	129.55 GB
DFS Used%:	0%
DFS Remaining%:	93.6%
Block Pool Used:	72 KB
Block Pool Used%:	0%
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	3 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	
Number of Under-Replicated Bl	
Number of Blocks Pending Dele	

- http://namenode:50070/dfshealth.html
- Live Nodes 가 연결된 datanode 수

하둡 2.0 관리

- **HDFS datanode 정보**

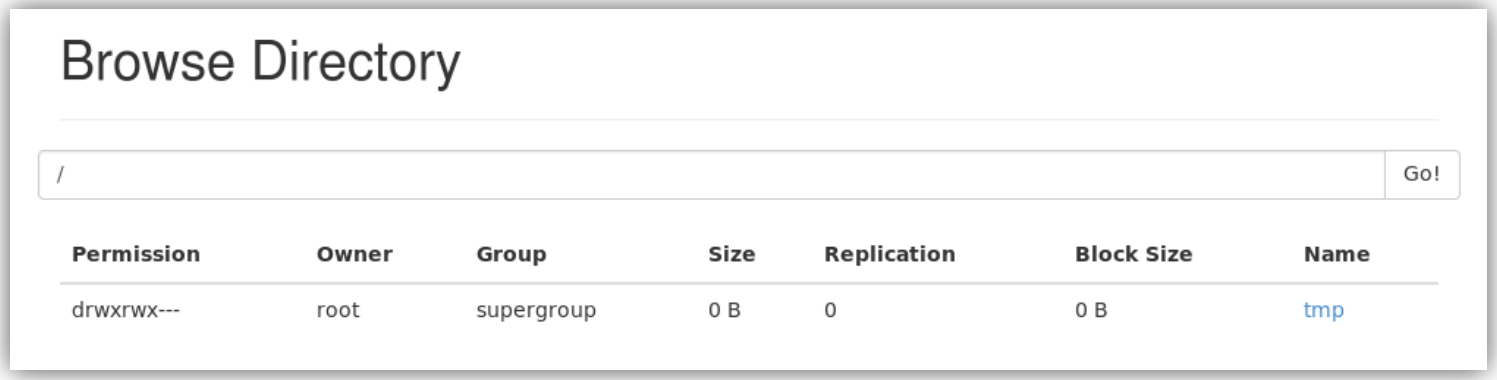
Datanode Information

In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
datanode2 (192.168.56.102:50010)	0	In Service	46.13 GB	24 KB	2.95 GB	43.18 GB	0	24 KB (0%)	0	2.5.2
datanode3 (192.168.56.103:50010)	0	In Service	46.13 GB	24 KB	2.95 GB	43.18 GB	0	24 KB (0%)	0	2.5.2
datanode1 (192.168.56.101:50010)	2	In Service	46.13 GB	24 KB	2.95 GB	43.18 GB	0	24 KB (0%)	0	2.5.2

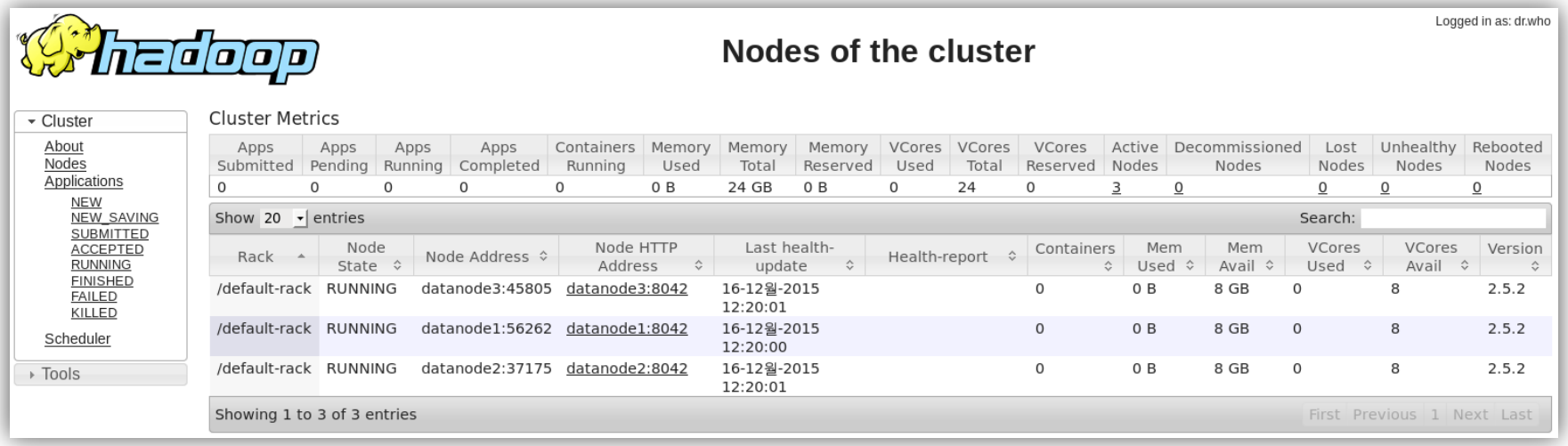
하둡 2.0 관리

- HDFS 디렉터리 정보



하둡 2.0 관리

- Yarn 클러스터 노드 정보



하둡 2.0 관리

• 분산파일 시스템(HDFS) 명령어

- hdfs dfs –리눅스 명령어 -옵션 명령행인자
- ex) hdfs dfs –mkdir –p /user/hadoop

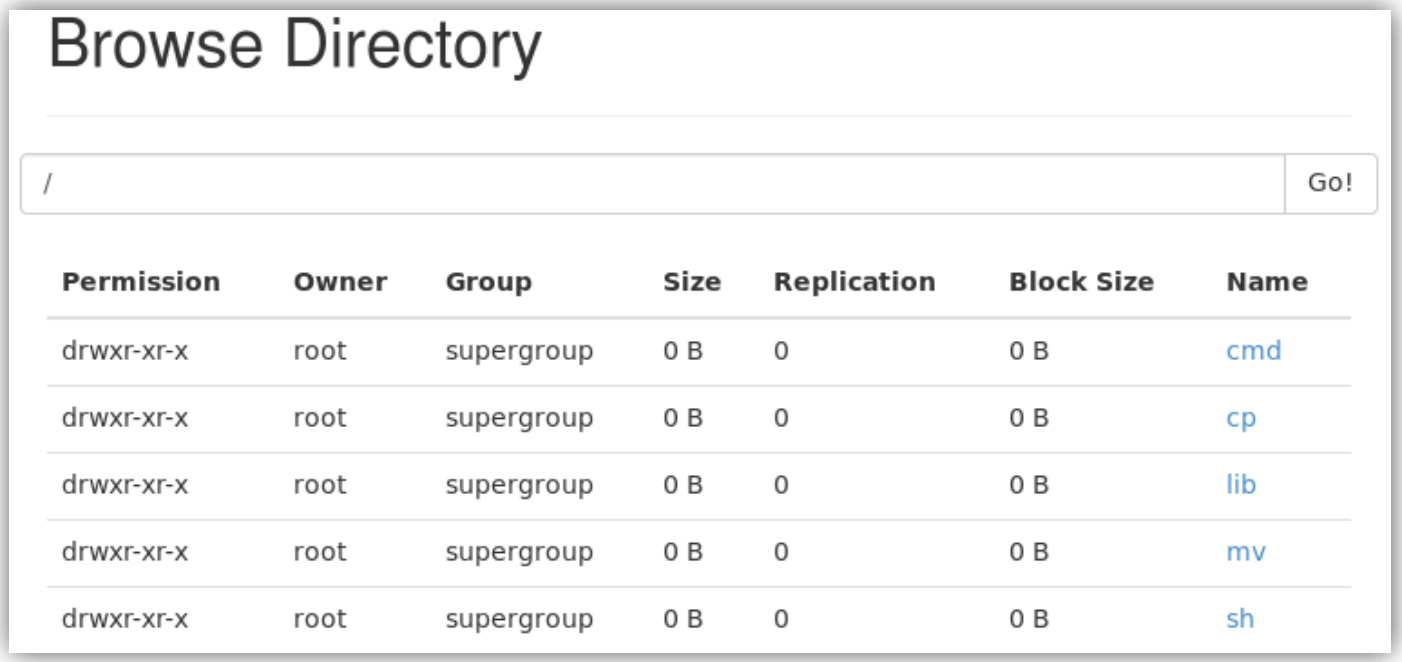
하둡 1.x에는 명령어에 옵션이 포함되는 형식이었다. 아래 두 명령은 동일하다.
2.x : hdfs dfs –ls –R /
1.X : hadoop fs –lsr /

- 명령어
  - du, du -s : 파일 용량 확인
  - expunge : 휴지통 비우기
  - mkdir, mkdir -p : 디렉터리 생성
  - ls, ls -R : 파일 또는 디렉터리 목록 보기
  - put, get : 파일 복사(로컬 <-> HDFS)
  - cat(text), text(text, zip) : 파일 내용 보기
  - cp, mv : 파일 복사, 이동(HDFS <-> HDFS)
  - chmod, chown, chgrp : 권한, 소유주, 그룹 변경
  - stat : 통계 정보 조회(포맷은 %b, %F, %n, %o, %r, %y, %Y)
  - rm, rm -R, rm -skipTrash : 파일 삭제, 디렉터리 삭제, 완전 삭제
  - test : 파일 형식 확인
  - setrep : 복제 수 변경
  - count : 카운트 값 조회
  - touchz : 0바이트 파일 생성
  - head : 파일의 처음 내용 확인
  - tail : 파일의 마지막 내용 확인

하둡 2.0 관리

- 디렉터리 생성 및 조회 명령

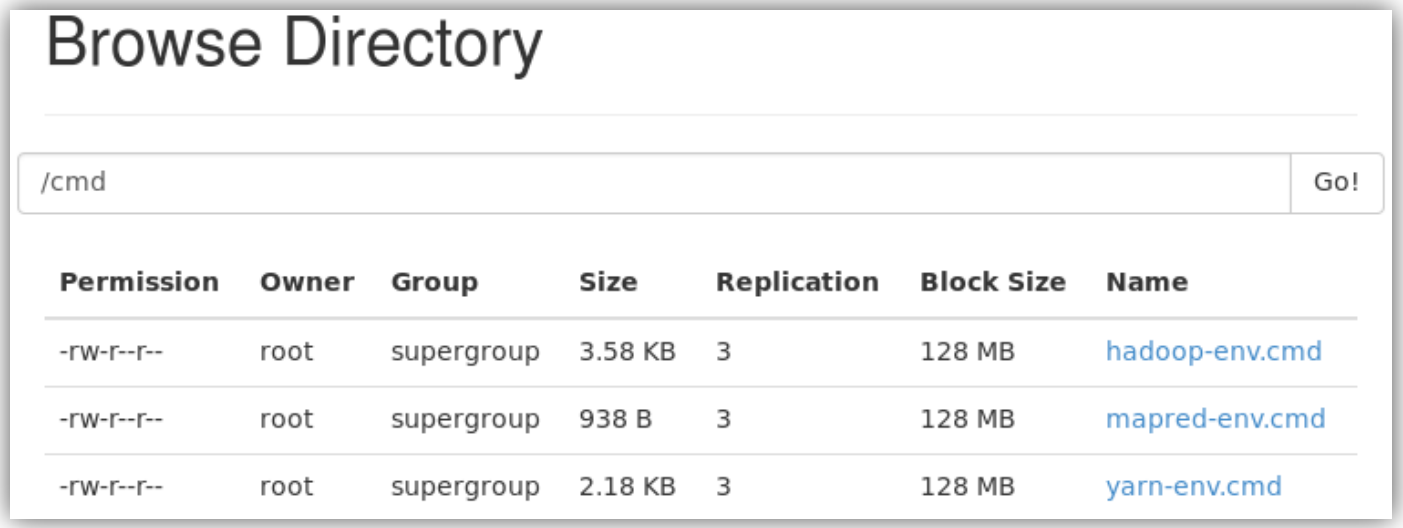
```
# cd $HADOOP_HOME/etc/hadoop
# hdfs dfs –mkdir  /sh /cmd /cp
# hdfs dfs -mkdir  /lib /mv
# hdfs dfs –ls / 또는 # hadoop fs -ls /(하둡 1.x 명령)
```



하둡 2.0 관리

- 파일 복사 명령어(로컬 <---> HDFS)

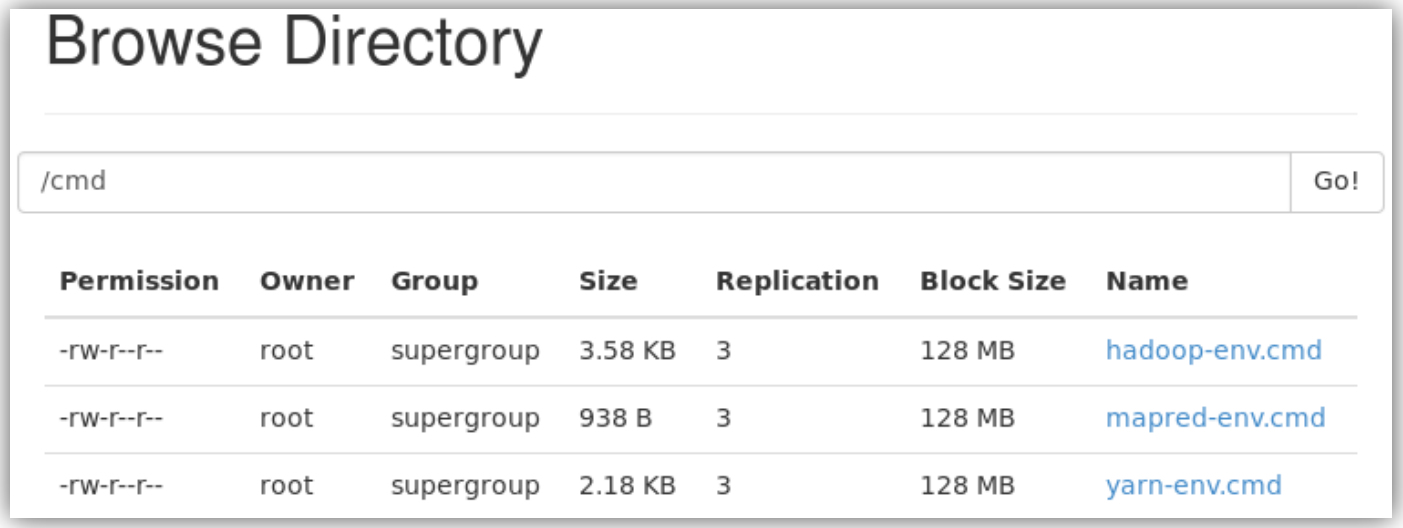
```
# cd $HADOOP_HOME/etc/hadoop
# hdfs dfs -put ./*.sh  /sh/,
# hdfs dfs -put ./*.cmd  /cmd/
# hdfs dfs -put ../../lib/* /lib/
```



하둡 2.0 관리

- 파일 복사 명령어(로컬 <---> HDFS)

```
# cd $HADOOP_HOME/etc/hadoop
# hdfs dfs -put ./*.sh  /sh/
# hdfs dfs -put ./*.cmd  /cmd/          # hdfs dfs -put ../../lib/* /lib/
```

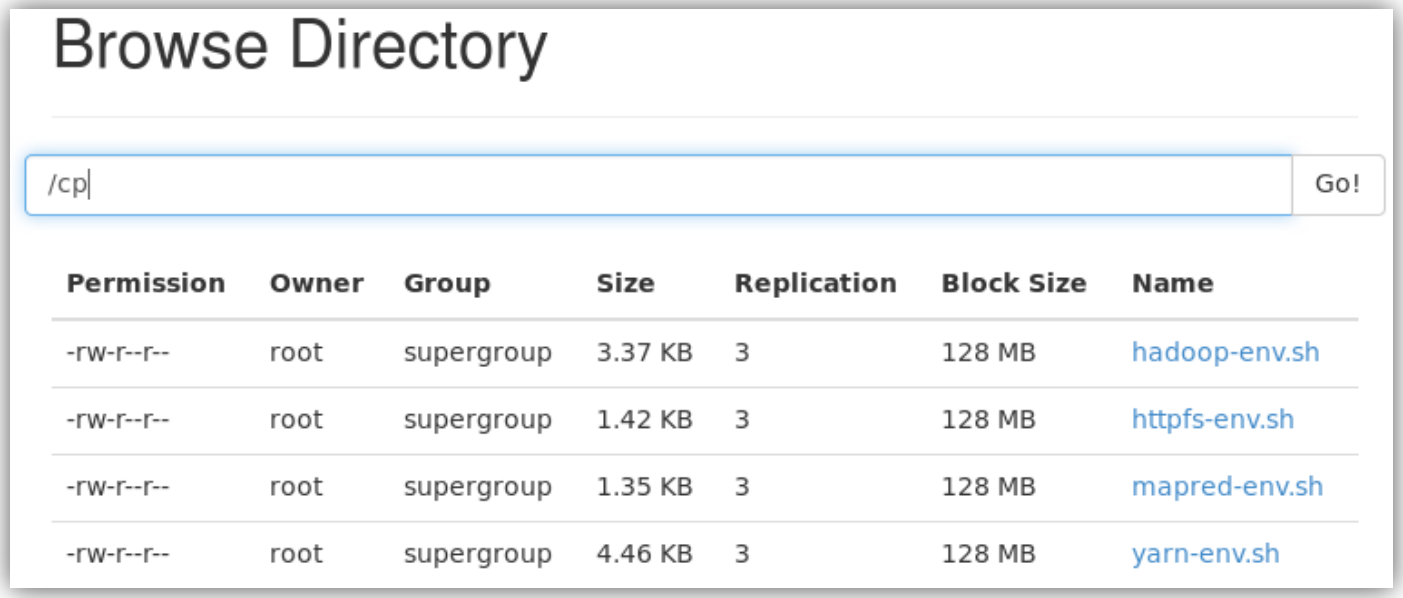



```
# cd /home/root   # mkdir hdfs_get   # hdfs dfs -get /sh/* hdfs_get/
# hdfs dfs -getmerge /cmd/ ./hdfs_get/cmd_merge.cmd     # ls hdfs_get
cmd_merge.cmd  hadoop-env.sh  httpfs-env.sh  mapred-env.sh  yarn-env.sh
```

하둡 2.0 관리

- 파일 복사, 이동, 삭제 명령어(HDFS <---> HDFS)

```
# hdfs dfs -cp  /sh/* /cp/
# hdfs dfs -mv  /sh/* /mv/
# hdfs dfs -rmr  /lib/*
```



분산 저장시 복사는 실제 물리적으로 저장되지만 이동은 메타 정보만 수정되기 때문에 이동 속도가 매우 빠르다.

하둡 2.0 관리

• 통계정보 조회 명령

stat 명령은 지정한 경로에 대한 통계 정보를 조회하는 명령

옵션	설 명
%b	블록단위 파일크기
%F	디렉터리면 directory 파일이면 regular fiel임
%n	디렉터리명 또는 파일명
%o	블록 크기
%r	복제 파일 개수
%y	갱신일자 yyyy-MM-dd HH:mm:ss
%Y	유닉스 타임스탬프 형식으로 출력

```
# hdfs dfs –stat %n%F%b%y /cmd/*
hadoop-env.cmdregular file36702015-12-15 14:12:27
mapred-env.cmdregular file9382015-12-15 14:12:27
yarn-env.cmdregular file22372015-12-15 14:12:28
```

하둡 2.0 관리

- **파일 처음과 마지막 내용 확인 명령**

먼저 기상청 강수량 자료 2006-2015.csv 파일을 호스트 컴퓨터 D:₩data 폴더에 복사하고 아래 명령을 이용해 마운트 하여 namenode로 복사하고 HDFS에 kma 디렉터리를 생성한다.

```
# cd /home/root   # mount –t vboxsf data data  # cp ./data/2006-2015.csv ./
# hdfs dfs -mkdir /kma       # hdfs dfs -put ./2006-2015.csv /kma/
# hdfs dfs -ls /kma

-rw-r--r--   3 root supergroup       84410 2015-12-16 00:27 /kma/2006-2015.csv
```

- 2006-2015.csv 파일의 처음 10라인 출력하기(한글은 깨져서 출력된다.)

```
# hdfs dfs -cat /kma/2006-2015.csv | head -10
```

- 2006-2015.csv 파일의 마지막 10라인 출력하기(한글은 깨져서 출력된다.)

```
# hdfs dfs -cat /kma/2006-2015.csv | tail -10 <- 권장하지 않음
# hdfs dfs -tail /kma/2006-2015.csv <- 파일의 마지막 1KB 출력
```

하둡 2.0 관리

- 데이터 노드 변경하기 실습 전 설정

DataNode 3개 -> 2개로 축소 후 datanode1, datanode2만으로 운영하고 datanode3는 노드 추가/삭제에 사용

1. /etc/hosts 파일 수정(기존 그대로 사용하면 됨)

namenode, 각 datanode 동일하게 설정해야 함

2. $HODOOP_HOME/etc/hadoop/slaves 파일을 아래와 같이 수정

```
# cd $HODOOP_HOME/etc/hadoop
# vi slaves
datanode1
datanode2
```

하둡 2.0 관리

- 데이터 노드 변경하기 실습 전 설정

3. hdfs-site.xml 아래와 같이 수정

```
# vi hdfs-site.xml
<configuration>
        <property>
                <name>dfs.replication</name>
                <value>2</value>
        </property>
</configuration>

# start-dfs.sh
```

※ 브라우저에서 HDFS 상태 확인

-. http://namenode:50070/dfshealth.html

하둡 2.0 관리

- 데이터 노드 상태

Configured Capacity:	92.27 GB
DFS Used:	11.27 MB
Non DFS Used:	5.9 GB
DFS Remaining:	86.35 GB
DFS Used%:	0.01%
DFS Remaining%:	93.59%
Block Pool Used:	11.27 MB
Block Pool Used%:	0.01%
DataNodes usages% (Min/Median/Max/stdDev):	0.01% / 0.01% / 0.01% / 0.00%
Live Nodes	2 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	
Number of Under-Replicated Blo	
Number of Blocks Pending Deleti	

- http://namenode:50070/dfshealth.html
- Live Nodes 가 연결된 datanode 수

하둡 2.0 관리

- 데이터 노드 추가

1. hdfs-site.xml 파일에서 dfs.hosts에 지정한 include.hosts 파일에 새롭게 추가될 datanode3를 추가한다.
2. 새로 추가된 datanode3에서 데이터 노드 프로세스를 시작한다.
3. Namenode에서 허가된 데이터 노드 집합을 반영한다.
   hdfs dfsadmin –refreshNodes
4. 새로 허가된 노드 매니저 집합을 리소스 매니저에 반영한다.
5. yarn rmadmin -refreshNodes
6. 새로운 데이터 노드가 하둡 제어 스크립트에 의해 지금 부터 클러스터에서 사용될 수 있도록 slaves 파일을 갱신한다.
7. 새로운 데이터 노드와 태스크 트래커가 웹 UI에 나타나는지 확인한다.

하둡 2.0 관리

- 데이터 노드 추가하기

```
# cd $HADOOP_HOME/etc/hadoop
# vi hdfs-site.xml
<configuration>
        <property>
                <name>dfs.replication</name>
                <value>2</value>
        </property>
        <property>
                <name>dfs.hosts</name>
                <value>/home/root/include.hosts</value>
        </property>
        <property>
                <name>dfs.hosts.exclude</name>
                <value>/home/root/exclude.hosts</value>
        </property>
</configuration>
```

하둡 2.0 관리

- 데이터 노드 추가하기

```
# cd /home/root/hadoop-2.5.2/etc/hadoop/
# vi include.hosts
datanode1
datanode2

# touch exclude.hosts
파일 내용 없음

# hdfs dfsadmin  –refreshNodes
```

하둡 2.0 관리

- 데이터노드 변경하기

```
# cd /home/root/
# vi include
datanode1 → 사용할 datanode
datanode2 → 사용하지 않을 datanode

# vi exclude
datanode3 → 사용하지 않을 datanode


# hdfs dfsadmin –refreshNodes
```

```
****** datanode4 생성

# scp ~/.ssh/* 192.168.56.104:~/.ssh/

 # vi /etc/hosts
192.168.56.104 datanode4    추가

# scp /etc/hosts datanode4:/etc/
# scp /etc/hosts datanode1:/etc/
# scp /etc/hosts datanode2:/etc/

# scp /etc/profile datanode4:/etc/
```

하둡 2.0 관리

- 데이터노드 추가

```
# cd /home/root
# scp -r hadoop-2.5.2 datanode4:/home/root/
# ssh datanode4 service iptables stop

# ssh datanode4
# vi /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=datanode4
```

----datanode4 재부팅

하둡 2.0 관리

- 데이터노드 변경

ssh datanode4 service iptables stop
namenode의 새로운 터미널에서..

cd /home/root
vi include
datanode4 추가

cd $HADOOP_HOME/etc/hadoop
vi hdfs-site.xml

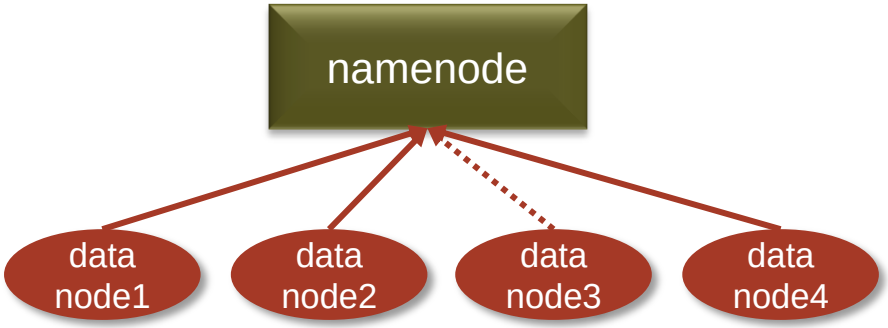
```
<configuration>
        <property>
                <name>dfs.replication</name>
                <value>4</value>
        </property>
</configuration>
```

하둡 2.0 관리

- 데이터노드 추가

```
# scp hdfs-site.xml datanode1:$HADOOP_HOME/etc/hadoop
# scp hdfs-site.xml datanode2:$HADOOP_HOME/etc/hadoop
# scp hdfs-site.xml datanode3:$HADOOP_HOME/etc/hadoop
# scp hdfs-site.xml datanode4:$HADOOP_HOME/etc/hadoop
```

```
# vi slaves
datanode4 추가
```




```
#stop-dfs.sh
#start-dfs.sh
# hadoop-daemon.sh start datanode
```

하둡 2.0 관리

- 네임노드 추가

*** namenode2 머신 추가
host name : namenode2
network ip : 192.168.56.200
netmask : 255.255.255.0
gateway : 192.168.56.1

vi slaves
datanode4 추가

#stop-dfs.sh
#start-dfs.sh
hadoop-daemon.sh start datanode

Chapter 06 R프로그래밍

- 기본 데이터형
- 변수와 함수
- 제어문
- 백터와 행렬
- 입출력 함수
- 문자열 처리 함수
- 그래픽 함수